UNIT V

DESIGN AND ANALYSIS OF MACHINE LEARNING EXPERIMENTS

Guidelines for machine learning experiments, Cross Validation (CV) and resampling – K-fold CV, bootstrapping, measuring classifier performance, assessing a single classification algorithm and comparing two classification algorithms – t test, McNemar's test, K-fold CV paired t test

5.1. GUIDELINES FOR MACHINE LEARNING EXPERIMENTS

Machine learning projects are highly iterative; as you progress through the ML lifecycle, you'll find yourself iterating on a section until reaching a satisfactory level of performance, then proceeding forward to the next task (which may be circling back to an even earlier step). Moreover, a project isn't complete after you ship the first version; you get feedback from real-world interactions and redefine the goals for the next iteration of deployment.

Machine learning experiments can take a long time. Hours, days, and even weeks in some cases. This gives you a lot of time to think and plan for additional experiments to perform. In addition, the average applied machine learning project may require tens to hundreds of discrete experiments in order to find a data preparation model and model configuration that gives good or great performance. The drawn-out nature of the experiments means that you need to carefully plan and manage the order and type of experiments that you run.

With this approach, you will be able to:

- Stay on top of the most important questions and findings in your project.
- Keep track of what experiments you have completed and would like to run.

- Zoom in on the data preparations, models, and model configurations that give the best performance.

5.1.1. PROJECT LIFE CYCLE

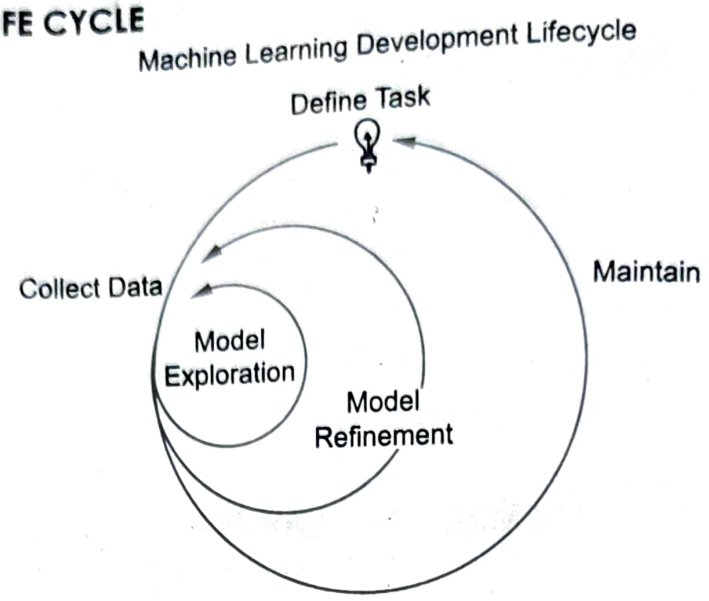


Fig. 5.1.

1. Planning and project setup

- Define the task and scope out requirements
- Determine project feasibility
- Discuss general model tradeoffs (accuracy vs speed)
- Set up project codebase

2. Data collection and labeling

- Define ground truth (create labeling documentation)
- Build data ingestion pipeline
- Validate quality of data

3. Model exploration

- Establish baselines for model performance
- Start with a simple model using initial data pipeline
- Overfit simple model to training data
- Stay nimble and try many parallel (isolated) ideas during early stages

- Find SoTA model for your problem domain (if available) and reproduce results, then apply to your dataset as a second baseline

4. Model refinement

- Perform model-specific optimizations (ie. hyperparameter tuning)
- Iteratively debug model as complexity is added
- Perform error analysis to uncover common failure modes

5. Testing and Evaluation

- Evaluate model on test distribution; understand differences between train and test set distributions (how is "data in the wild" different than what you trained on)
- Revisit model evaluation metric; ensure that this metric drives desirable downstream user behavior
- Write tests for: input data pipeline, model inference functionality, model inference performance on validation data, explicit scenarios expected in production.

6. Model deployment

- Expose model via a REST API
- Deploy new model to small subset of users to ensure everything goes smoothly, then roll out to all users
- Maintain the ability to roll back model to previous versions
- Monitor live data and model prediction distributions

7. Ongoing model maintenance

- Understand that changes can affect the system in unexpected ways
- Periodically retrain model to prevent model staleness

5.1.2. ANATOMY OF A MACHINE LEARNING EXPERIMENT

The key challenge with tracking machine learning experiments is that there are too many entities to track and complex relationships between them. Entities include parameters, artifacts, jobs and relationships could be one-to-one, one-to-many, many-to-one between experiments, trials and entities.

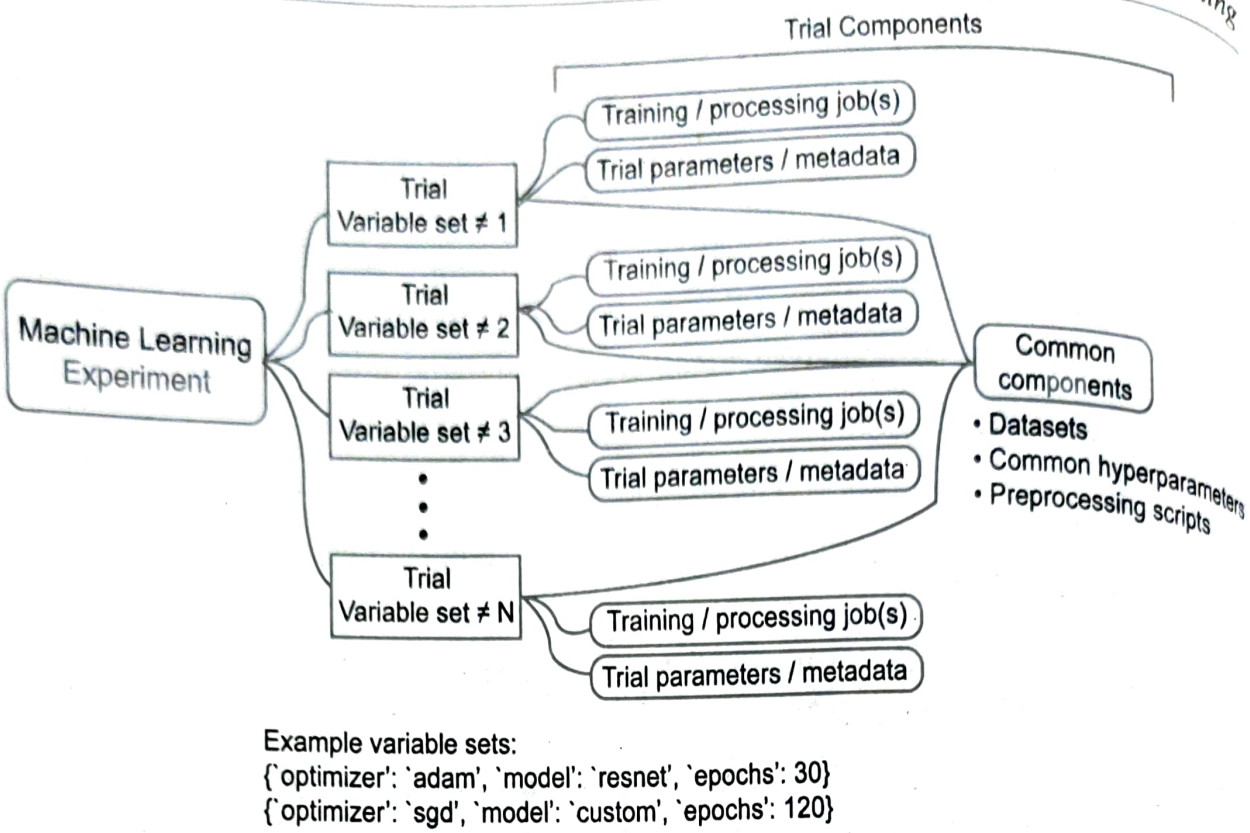


Fig. 5.2.

5.1.3. PROPERTIES OF AN EXPERIMENT, TRIAL AND TRIAL COMPONENT

- An Experiment is uniquely characterized by its objective or hypothesis
- An Experiment usually contains more than one Trial, one Trial for each variable set.
- A Trial is uniquely characterized by its variable set, sampled from the variable space defined by you.
- A Trial component is any artifact, parameter or job that is associated with a specific Trial.
- A Trial component is usually part of a Trial, but it can exist independent of an experiment or trial.
- A Trial component cannot be directly associated with an Experiment. It has to be associated with a Trial which is associated with an Experiment.
- A Trial component can be associated with multiple Trials. This is useful to track datasets, parameters and metadata that is common across all Trials in an Experiment.

5.2. CROSS VALIDATION (CV) AND RESAMPLING

5.2.1. CROSS-VALIDATION

The main point in the Machine Learning modelling is a training of model. Basically, we build a model on the train data, the data that we have in hand. And we are unsure about the result of our model on unseen data if we have not test data additionally to check the model accuracy. Only train accuracy is not good estimate of the test accuracy, because after one point (optimal point) train score tends to decrease and it leads to overfitting. It means that while the flexibility of our model increase , model try to predict each observation in the train set exactly , and in this scenario we get almost 100 percent of train accuracy.

Our main aim is getting the optimal model which provide lower test error. To reach our goal, we should split our data into train and test set and always check the accuracy on our own test set.(or sets). This method is cross-validation and there are some types of cross validation as : Validation set Approach, Leave One Out Cross Validation(LOOCV), and k-fold cross-validation.

5.2.2. THE VALIDATION SET APPROACH

Validation Set approach is very simple method and frequently used method when there is sufficiently enough amount of observations to get reasonable results. It is basically dividing the data that we have to two place as train set and validation set (or holdout set), and building model on train set , then checking the model accuracy on validation set. And resulting accuracy from validation set is the estimate about the real test data(unseen data).

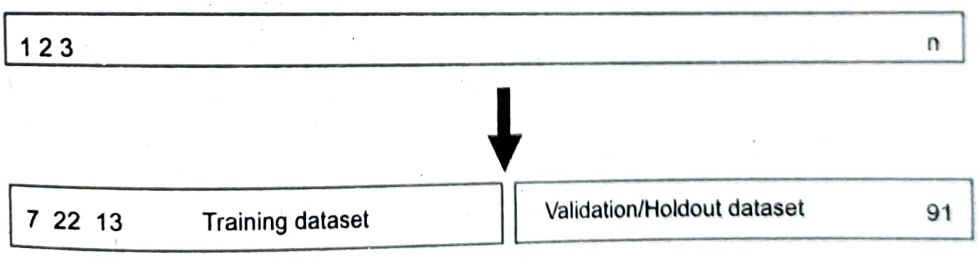


Fig. 5.3.

But there are some drawbacks about this method. Firstly, we split the data set randomly to train and validation parts and fit our model on those train observations. So our model trains on only those observations and maybe the observations in validation set are easy to predict, and this lead to high variability in the accuracy,

especially in small data sets. If we choose ten times different subsets as train and validation set and check the accuracy, we can see that it is not always same.

And also we train our model on the some part of observations only , and it leads to if we have not enough amount of data it can be overestimate the real accuracy. For example KNN can be best classifier for our problem but as we train on the small set, our validation set accuracy will be lower and we will think that it is not appropriate model, but in fact it is.

Our data has 300 observations:

```
data = df[0:300]
data.shape
```

```
(300, 10)
```

And we will fit and check accuracy of model 50 times with Validation Set Approach:

```
trial = 50
scores = []
for i in range(trial):
    x = data[features]
    y = data["Survived"]
    X_train,X_test, y_train,y_test = train_test_split(x,y,test_size = 20)
    model = svm.SVC(C=1)
    model.fit(X_train,y_train)
    score = model.score(X_test,y_test)
    scores.append(score)
plt.plot(scores)
plt.xlabel("Number of trial")
plt.ylabel("Test Accyracy Estimation")
plt.show()
```

We can see how scores differ during trials:

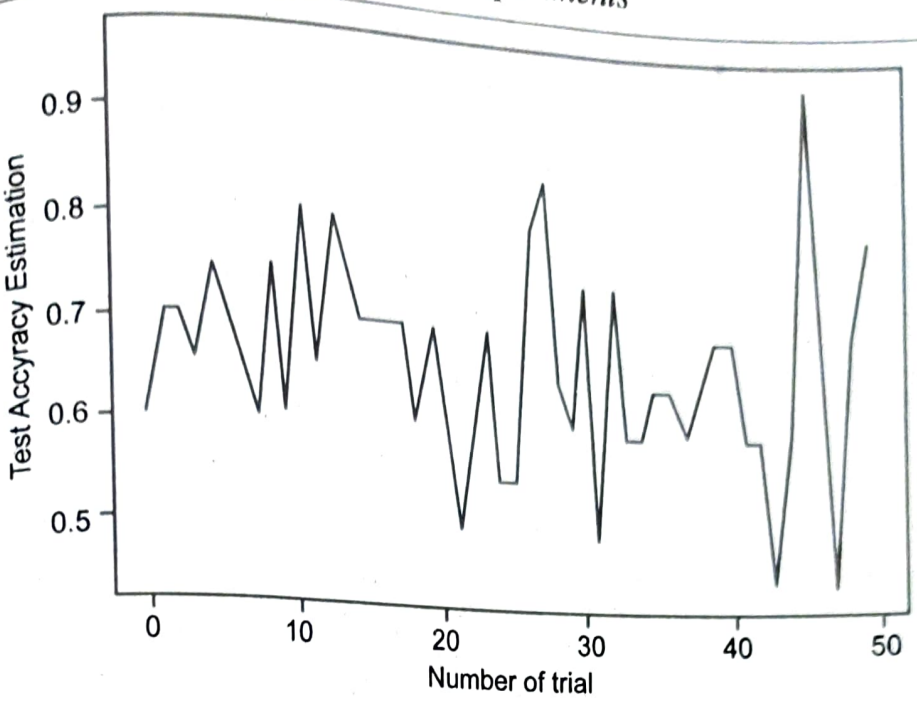


Fig. 5.4.

LOOCV:

Leave One Out Cross Validation method is addressed to the drawbacks of the validation set approach and it is also simple method as validation set approach. Main point here is to take each observation as a validation set one time. It means that if we have "n" number of observations, we will fit model "n" times. And in every try we will keep one observation as a test sample, and will train the model on "$n - 1$" observations. At the end we will have "n" accuracy scores, and we can take the mean of these scores as the overall score of the model.

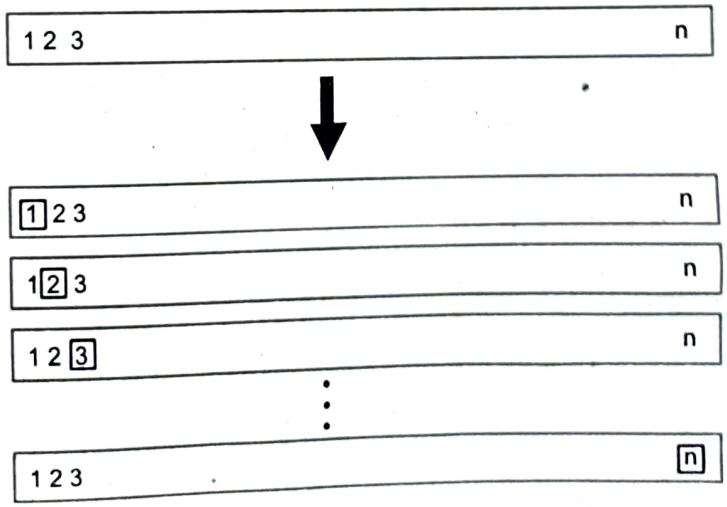


Fig. 5.5.

As in this method each observation becomes test sample one time, it is highly unbiased method for estimating the real accuracy. But as we said bias is not the only case, and there should be also low variance. In LOOCV variance is high, because we test our model every time on single observation and it is not enough to get more accurate estimations. Also it is computationally expensive, as we fit model "n" times. For example, if we have 1000 observations, it means that we will fit 1000 models.

5.3. K-FOLD CROSS-VALIDATION

Most used cross-validation technique is k-Fold method. Here the procedure is actually same with LOOCV but we do not fit model "n" times, "K" is the number of folds, for example 5-Fold cross-validation. Lets say that we have 100 observations and we will use 5-Fold cross validation. It means that we will fit model only five times. In each iteration we will keep 20 observation as a validation set and 80 observation as train. As in the picture below:

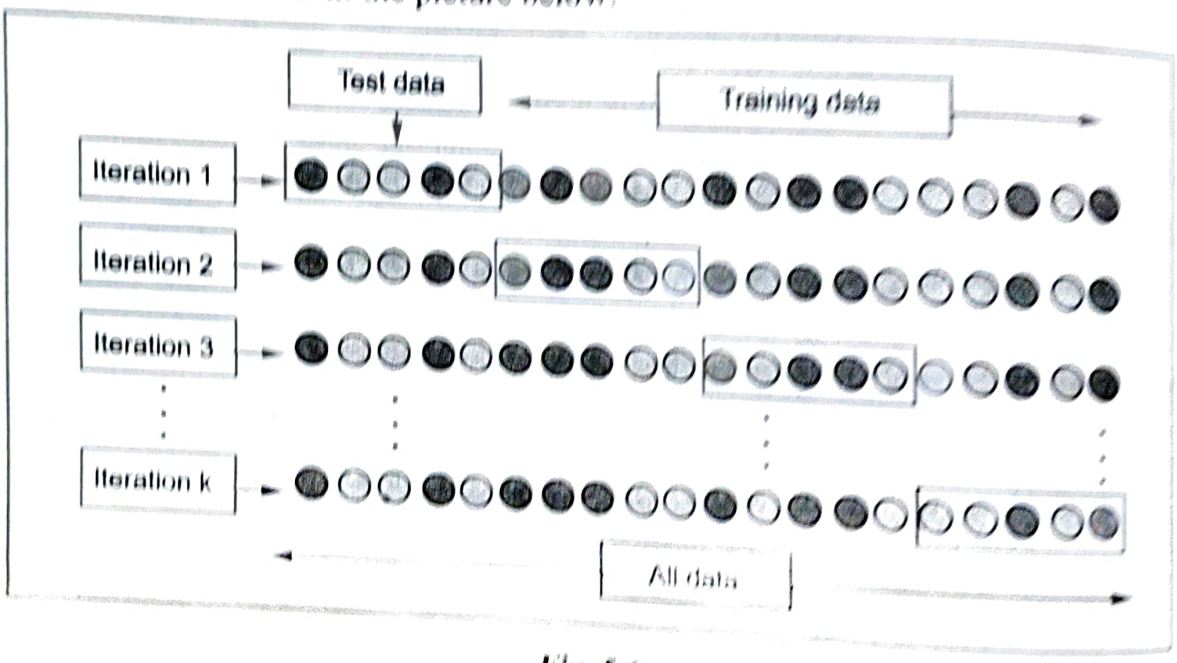


Fig. 5.6.

After 5 iteration, we will have 5 score and we can take the mean of these scores as overall estimation of accuracy. This method is optimal method, bias is not high as validation set approach and variance is not high as LOOCV. And also is not so computationally expensive. In practise frequently used ones are 5-Fold and 10-Fold.We can use our data for modelling and estimate the accuracy with 10-Fold method 50 times;

```
trial = 50
k_fold_scores = []
for i in range(trial):
    x = data[features]
    y = data["Survived"]
    model = svm.SVC(C=1)
    score_k_fold = (cross_val_score(model, x, y, cv=10)).mean()
    k_fold_scores.append(score_k_fold)
plt.plot(k_fold_scores)
plt.xlabel("Number of trial")
plt.ylabel("Test Accyracy Estimation")
plt.show()
```

We can see that score estimation is constant and is about 64, so we can say that our model is not good choice for this problem. But in Validation set approach, we can see that our test accuracy is over 80 by chance because of random splitting and think that it is good model:

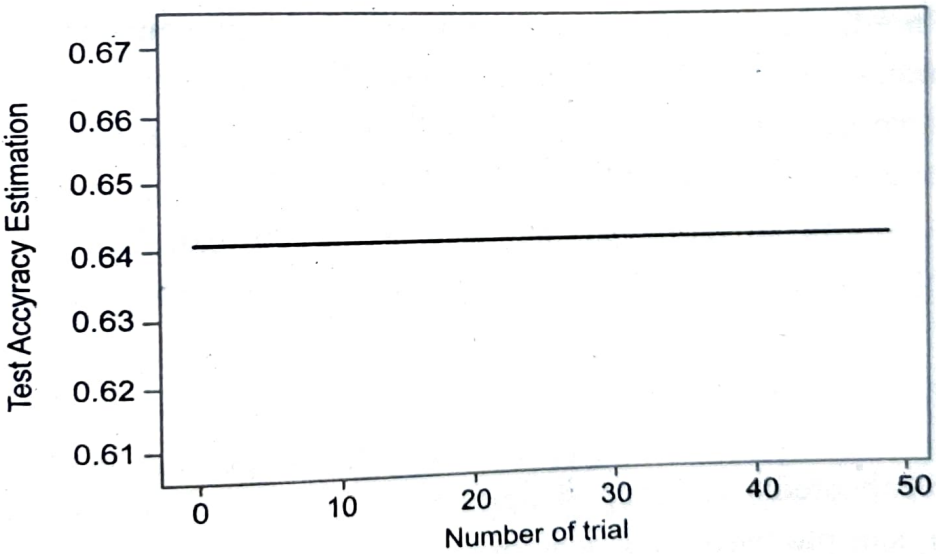


Fig. 5.7.

And real test score is: 0.58. We can see that 5-Fold validation provides more accurate estimations for accuracy on unseen data

```
#Accuracy on Real unseen data
real_test = df[301:500]

x = data[features]
y = data["Survived"]

x_r = real_test[features]
y_r = real_test["Survived"]
final_model = svm.SVC(C=1)
final_model.fit(x,y)
final_model.score(x_r,y_r)


0.5829145728643216
```

In conclusion, based on the size of the data we have, we can use sampling and model selection methods above to estimate accuracy of models, parameter tuning, and so on.

5.4. RESAMPLING METHODS

Resampling methods are very useful and beneficial in statistics and machine learning to fit more accurate models, model selection and parameter tuning. They draw samples from train data and fit model to check the variability of model and get additional information. We cannot best sure of the result of the model by just unique fit without testing on different sample or samples. It can be computationally expensive because of fitting model more than one, but recent improvements tackle this issue easily without too much effort.

For example, cross-validation is effective method to estimate the error of model on unseen data, to determine the flexibility of model, parameter selection and so on.It is a process of repeatedly drawing samples from a data set and refitting a given model on each sample with the goal of learning more about the fitted model. Resampling methods can be expensive since they require repeatedly performing the same statistical methods on N different subsets of the data.

Following are types of resampling methods:

- Bootstrap Sampling

- K Fold Cross Validation
- Leave One Out Cross Validation

5.4.1. BOOTSTRAP SAMPLING METHOD

- Configure bootstrap
- Run bootstrap in train, test data and obtain accuracy scores
- Visual representation using Histogram and derive Confidence levels

Here a random function is used to create samples from original data. Within a sample set, there could be duplicates or more however 2 sample sets are unlikely to be 100% same. Due to the drawing with replacement, a bootstrapped data set may contain multiple instances of the same original cases, and may completely omit other original cases.

This technique is used in machine learning to estimate the skill of machine learning models when making predictions on data not included in the training set. This technique helps to fine tune the model even before we give it access to test data (real world data). This allows us to tweak the model hyperparameters to achieve the best score.

Bootstrapping is primarily used to establish empirical distribution functions for a widespread range of statistics. A desirable property of the results from estimating machine learning model skills is that the estimated skill can be presented with confidence intervals, a feature not readily available with other methods such as Cross Validation.

In [2]:

```
data = pd.read_csv('../input/handson-pima/Hands on Exercise Feature Engineering_
pima-indians-diabetes (1).csv')
data.head()
values = data.values
```

In [3]:

```
#Lets configure Bootstrap
n_iterations = 10  #No. of bootstrap samples to be repeated (created)
n_size = int(len(data) * 0.50) #Size of sample, picking only 50% of the given data in
every bootstrap sample
```

In [4]:

```
#Lets run Bootstrap
stats = list()
for i in range(n_iterations):

   #prepare train & test sets
   train = resample(values, n_samples = n_size) #Sampling with replacement..whichever is not used in training data will be used in test data
   test = np.array([x for x in values if x.tolist() not in train.tolist()]) #picking rest of the data not considered in training sample
      #fit model
   model = DecisionTreeClassifier()
   model.fit(train[:,:-1], train[:,-1]) #model.fit(X_train,y_train) i.e model.fit(train set, train label as it is a classifier)

   #evaluate model
   predictions = model.predict(test[:,:-1]) #model.predict(X_test)
   score = accuracy_score(test[:,-1], predictions) #accuracy_score(y_test, y_pred)
   #caution, overall accuracy score can mislead when classes are imbalanced

   print(score)
   stats.append(score)
```

```
0.655982905982906
0.7130801687763713
0.6631130063965884
0.6802575107296137
0.7021276595744681
0.6919831223628692
```

0.6886993603411514

0.6708860759493671

0.6702355460385439

0.6865671641791045

Here each Bootstrap iteration sample would create one model and this model is tested against the Out of Bag (test data) of that sample, i.e we will test that sample with the test sample not part of that sample.

Thus we obtain accuracy scores for 10 samples.

5.5. CROSS VALIDATION - KFOLD

Cross validation resamples without replacement and thus produces surrogate data sets that are smaller than the original. These data sets are produced in a systematic way so that after a pre-specified number k of surrogate data sets, each of the n original cases has been left out exactly once. This is called k-fold cross validation or leave-x-out cross validation with $x = n / k$, e.g. leave-one-out cross validation omits 1 case for each surrogate set, i.e. $k = n$.

Primary purpose of CV is measuring performance of a model

- Fit Logistic Regression and compute cross_val_score
- Calculate accuracy of this model

In [7]:

```
#Create separate arrays such that only values are considered as X, y
values = data.values
X = values[:,0:8]
y = values[:,8]

#Split the data into train,test set
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.50, random_state
= 1)
```

In [8]:

```
#Lets configure Cross Validation
```

```
#default value of n_splits = 10
kfold = KFold(n_splits = 50, random_state = 7)
model = LogisticRegression()
results = cross_val_score(model,X,y,cv = kfold)
```

In [9]:

```
print(results)
```

```
[0.75       0.625      0.5625     0.75       0.8125     0.8125
 0.875      0.75       0.8125     0.8125     0.8125     0.75
 0.8125     0.5        0.9375     0.6875     0.75       0.5625
 0.6        0.66666667 0.66666667 0.8        0.8        0.86666667
 0.8        0.66666667 0.8        0.86666667 0.73333333 0.86666667
 0.8        0.66666667 0.8        1.         0.73333333 0.86666667
 0.73333333 0.8        0.93333333 1.         0.73333333 0.73333333
 0.73333333 0.6        0.86666667 0.66666667 0.8        0.86666667
 0.86666667 0.8       ]
```

Since 50 *n*_folds is requested hence you received 50 iterations of this test set. For 50 test sets, 50 training sets are created and it gives 50 different accuracy scores.

In [10]:

```
#What's the accuracy of this model using KFold CV


print('Accuracy:%.3f%% (%.3f%%)' % (results.mean()*100.0,
results.std()*100.0))
```

Accuracy: 77.017% (10.621%)

Cross Validation - Leave One Out CV

Here the dataset k is split into k-1 train sets and 1 test set.

- ❖ Provide a dataset with values
- ❖ Fit LOOCV and print the train, test set values
- ❖ Calculate accuracy of this model

In [11]:

```
# scikit-learn k-fold cross-validation
# data sample
```

```
data = array([10,20,30,40,50,60,70,80,90,100])
# prepare cross validation
loocv = LeaveOneOut()
# enumerate splits
for train, test in loocv.split(data):
  print('train: %s, test: %s' % (data[train], data[test]))
```

```
train: [ 20  30  40  50  60  70  80  90 100], test: [10]
train: [ 10  30  40  50  60  70  80  90 100], test: [20]
train: [ 10  20  40  50  60  70  80  90 100], test: [30]
train: [ 10  20  30  50  60  70  80  90 100], test: [40]
train: [ 10  20  30  40  60  70  80  90 100], test: [50]
train: [ 10  20  30  40  50  70  80  90 100], test: [60]
train: [ 10  20  30  40  50  60  80  90 100], test: [70]
train: [ 10  20  30  40  50  60  70  90 100], test: [80]
train: [ 10  20  30  40  50  60  70  80 100], test: [90]
train: [10 20 30 40 50 60 70 80 90], test: [100]
```

Here the data comprises of an array from 10 - 100, LOOCV will leave one of these values out as a test value.

K-fold CV, bootstrapping

Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

5.5.1. WHAT IS K-FOLD CROSS VALIDATION?

K-Fold CV is where a given data set is split into a K number of sections/folds where each fold is used as a testing set at some point. Lets take the scenario of 5-Fold cross validation(K = 5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the

model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.

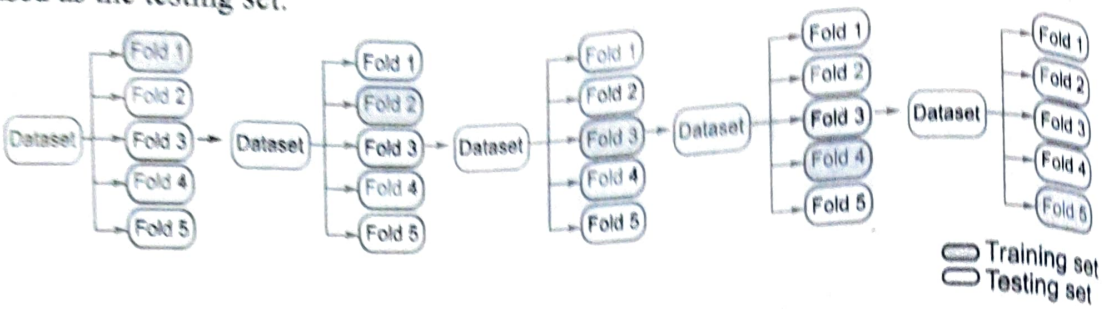


Fig. 5.8.

5.5.2. EVALUATING A ML MODEL USING K-FOLD CV

Lets evaluate a simple regression model using K-Fold CV. In this example, we will be performing 10-Fold cross validation using the RBF kernel of the SVR model

1. Importing libraries

First, lets import the libraries needed to perform K-Fold CV on a simple ML model.

```
import pandas
from sklearn.model_selection import KFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
import numpy as np
```

Lets see what we have imported,

pandas — Allows easy manipulation of data structures.

numpy — Allows scientific computing.

sklearn — A machine learning library for python.

2. Reading the data set

Now, lets read the data set we will be using, to a pandas data frame.

```
dataset = pandas.read_csv('housing.csv')
```

We will be using the Boston House price data set which has 506 records, for this example.

3. Pre-processing

We will now specify the features and the output variable of our data set.

```
X = dataset.iloc[:, [0, 12]]
y = dataset.iloc[:, 13]
```

The above code indicates that all the rows of column index 0-12 are considered as features and the column with the index 13 to be the dependent variable A.K.A the output. Now, lets apply the MinMax scaling pre-processing technique to normalize the data set.

```
scaler = MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)
```

This technique re-scales the data between a specified range(in this case, between 0–1), to ensure that certain features do not affect the final prediction more than the other features.

4. K-Fold CV

Now, lets get down to business.

```
scores = []
best_svr = SVR(kernel='rbf')
cv = KFold(n_splits=10, random_state=42, shuffle=False)
for train_index, test_index in cv.split(X):
    print("Train Index: ", train_index, "\n")
    print("Test Index: ", test_index)
    X_train, X_test, y_train, y_test = X[train_index], X[test_index], y[train_index],
y[test_index]
    best_svr.fit(X_train, y_train)  scores.append(best_svr.score(X_test, y_test))
```

We are using the RBF kernel of the SVR model, implemented using the sklearn library (the default parameter values are used as the purpose of this article is to show how K-Fold cross validation works), for the evaluation purpose of this example. First, we indicate the number of folds we want our data set to be split into. Here, we have used 10-Fold CV (n_splits=10), where the data will be split into 10 folds.

5.5.3. ADVANTAGES OF K-FOLD CROSS-VALIDATION

When we split a dataset into just one training set and one testing set, the test MSE calculated on the observations in the testing set can vary greatly depending on which observations were used in the training and testing sets.

By using k-fold cross-validation, we're able to use calculate the test MSE using several different variations of training and testing sets. This makes it much more likely for us to obtain an unbiased estimate of the test MSE.

K-fold cross-validation also offers a computational advantage over leave-one-out cross-validation (LOOCV) because it only has to fit a model k times as opposed to n times.

For models that take a long time to fit, k-fold cross-validation can compute the test MSE much quicker than LOOCV and in many cases the test MSE calculated by each approach will be quite similar if you use a sufficient number of folds.

5.6. BOOTSTRAPPING

Bootstrapping is a resampling method that is used in machine learning. It is a widespread technique due to its flexibility since it does not require anything other than your training dataset. Bootstrap Sampling comes from the ideas around just the Boostrap. The Bootstrap is a flexible and powerful statistical tool that brings us closer to our sample's true population parameters.

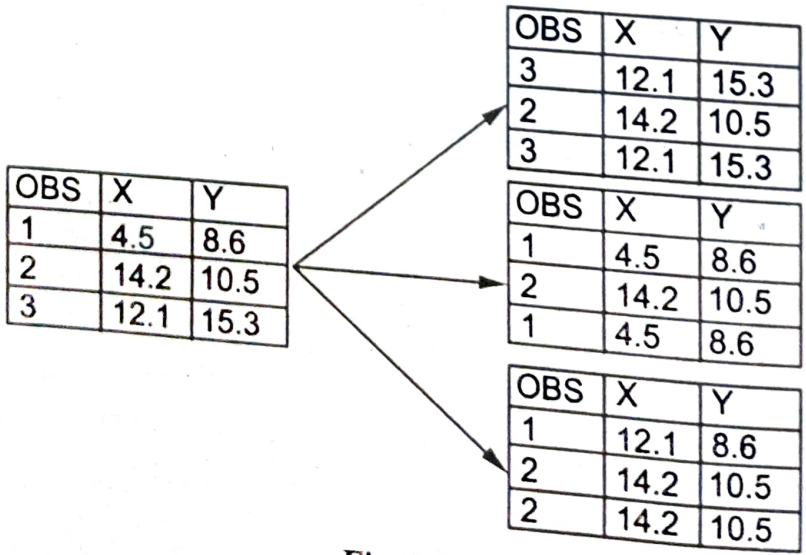


Fig. 5.9.

Bootstrap sampling is a type of resampling where we create N datasets from our population (your dataset) with replacement. Each bootstrap data set is the same size as our original dataset. As a result, some observations may appear more than once in a given bootstrap data set – and some not at all.

Here's an image that illustrates this idea, with $n = 3$ observations

5.6.1. IMPLEMENTING BOOTSTRAP SAMPLING IN PYTHON PANDAS

```
import pandas as pd
# import our dataset, first 100 rows
df = pd.read_csv('cars.csv')[0:100]
# create an array to store all of our dataframes
dataframes = []
# we will run this loop the amount of times that we have
# lines of data, in our case, 100 times
for i in range(len(df)):
    # append each of these data frames to our array
    # we set the length to be the original length of
    # our dataframe, and turn on replacement
    dataframes.append(df.sample(n=len(df), replace=True))
print([df.head() for df in dataframes])
```

(	Manufacturer–name	Model–name	Transmission	Color	Odometer–value
25	Subaru	Tribeca	automatic	other	250
38	Subaru	Outback	automatic	blue	3
91	Subaru	Forester	mechanical	blue	a few
62	Subaru	Outback	automatic	brown	
70	Subaru	Outback	mechanical		

allow our model to
his happens by resampling

	Year_produced	engine_fuel	engine_has_gas	engine_type	engine_capacity
25	2007	gasoline	False	gasoline	3.0
38	1999	gasoline	False	gasoline	2.5
91	2010	gasoline	False	gasoline	2.5
62	2008	gasoline	False	gasoline	2.5
70	2001	gasoline	False	gasoline	2.5

	...	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6
25	...	True	False	False	True	True	False
38	...	False	False	False	False	False	False
91	...	True	True	False	True	True	True
62	...	True	False	False	False	False	False
70	...	False	False	False	False	False	False

	feature_7	feature_8	feature_9	duration_listed
25	False	True	True	204
38	False	False	False	70
91	True	True	True	30
62	False	True	True	90
70	False	False	False	11

(5 rows × 30 columns)	Manufacturer_name	Model_name	Color	Transmission
ıbaru	Forester	mechanical	black	260000
ru	Legacy	mechanical	black	270000
	Legacy	mechanical	black	270000
	Forester	automatic	blue	242000
	XV	automatic	orange	183000

Year_produced	engine_fuel	engine_has_gas	engine_type	engine_capacity
1999	gasoline	False	gasoline	2.5
2004	gasoline	False	gasoline	2.0
2004	gasoline	False	gasoline	2.0
2003	gasoline	False	gasoline	2.5
2012	gasoline	False	gasoline	2.0

	...	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6
55	...	True	False	False	False	False	False
29	...	True	True	False	False	False	False
29	...	True	True	False	False	False	False
45	...	False	False	False	False	False	False
59	...	False	True	False	True	True	True

	feature_7	feature_8	feature_9	duration_listed
55	False	False	False	4
29	False	False	True	48
29	False	False	True	48
45	False	False	False	10
59	True	True	True	91

5.6.2. ADVANTAGES OF BOOTSTRAPPING IN MACHINE LEARNING

Bootstrapping has tons of advantages in Machine Learning. Here are a few

Improve Model Real-World Accuracy

Since we will create a lot more data, bootstrapping will allow our model to generalize to the underlying population. We now know this happens by resampling

your data with replacement, which means some data points will be repeated in the new dataset – moving us closer and closer to the true underlying distribution of our population. This reduces the variance of your estimates, leading to more accurate predictions and models with higher production success.

Increase Training Data Size

Let's be honest; sometimes, we're given datasets with insufficient data. This is one of the main advantages of bootstrap sampling. We can take a dataset with extremely high variance due to our low number of data points and perform our sampling. This will increase our dataset's bias and training size, leading to a model that can converge and provide accurate insights.

Disadvantages of Bootstrapping In Machine Learning

There are some disadvantages to bootstrapping should be considered before using this method.

Independent Population

When sampling with replacement, there is an underlying assumption that your data points are independent. Some data, like time series data, violates this, and the traditional bootstrap sampling method would not work.

Computational Limits

Since bootstrap sampling will create N new datasets, it is sometimes impossible to fit them into RAM. If our original dataset were 5,000 rows of data, our bootstrap sample would create 5,000 new datasets. While this parameter can be lowered, there is a computational and memory cap on bootstrap sampling that many run into.

What is the Difference Between Bootstrapping and Cross-Validation?

Bootstrapping and Cross-Validation are both sampling methods of statistical inference.

In general, statistical inference uses data from a sample to make estimates or predictions about a population. Both bootstrapping and cross-validation are used to estimate the performance of our population's "standard error" or, more simply, how our machine-learning algorithm will do in a production system on unseen data.

The main difference between the two methods is that bootstrapping is a resampling technique, while cross-validation is a partitioning technique.

ootstrapping involves random sampling with replacement from the training data set create multiple new training sets.

This means that bootstrapping will lower the variance for our machine-learning odel. This is great in situations where we are overfitting or want to increase our ias – but detrimental in situations where our variance is already low.

Cross-validation involves partitioning the training data set into multiple subsets nd training the algorithm on each subset. The performance of the algorithm is then valuated on a held-out test set. This means that cross-validation will take full dvantage of our dataset, but without the resampling techniques from bootstrapping, it is confined to only training on the samples we possess.

5.7. MEASURING CLASSIFIER PERFORMANCE

Classification is one of the most common tasks in machine learning. This is the case where the dependent variable only has discrete values. For example, imagine we have a dataset consisting of photos. Each photo is of a dog or a cat. As such, the task of assigning the label of 'dog' or 'cat' to each photo is a classification problem.

Before diving into the details, let's define some terms that will be useful. For simplicity, consider the case where only two classes are present in some data, 0 and 1. Assume we have a trained classifier. Furthermore, assume some test data is available where we know the true classes:

True Positives (TPTP) are instances where the classifier predicts a 1, and the true value is 1

False Positives (FPFP) are instances where the classifier predicts a 1, and the true value is 0

False Negatives (FNFN) are instances where the classifier predicts a 0, and the true value is 1

True Negatives (TNTN) are instances and the classifier predicts a 0, where the true value is 0

Accuracy

Accuracy measures the fraction of correctly classified samples. The following equation defines this value:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}\text{Accuracy} = TP + FP + TN + FNTP + TN \quad (1)$$

Precision

Precision measures the ability of the classifier to correctly label positive values. The following equation defines this value:

$$\text{Precision} = \frac{TP}{TP + FP}\text{Precision} = TP + FPTP \quad (2)$$

Recall

Recall measures the ability of the model to find all positive values. The following equation defines this value:

$$\text{Recall} = \frac{TP}{TP + FN}\text{Recall} = TP + FNTP \quad (3)$$

F1 Score

The F1 score is a weighted average of the precision and recall metrics. The following equation defines this value:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}F1 = \text{Precision} + \text{Recall}2 \times \text{Precision} \times \text{Recall} \quad (4)$$

Note that for all of the metrics above, the best possible outcome is 1.0. The worst possible result is a 0.0. Next let's cover the two plotting techniques I mentioned earlier.

ROC Plot

We can measure the diagnostic ability of a binary classifier by using the Receiver Operating Characteristic (ROC) plot. The raw output from a classifier are probabilities for each class in the dataset. We can determine the actual classes ('0' or '1') by using a threshold value to convert these probabilities. Normally, this threshold is set to thres = 0.5thres = 0.5. Therefore, anything less than thresthres is assigned '0', and the remainder are assigned '1'. For the ROC plot, the value of thresthres is varied between 0.00.0 and 1.01.0, and the performance of the model is illustrated in terms of the true positive rate (TPRTPR) and false positive rate (FPRFPR).

Definitions for these quantities include:

$$TPR = \frac{TP}{TP + FN}TPR = TP + FNTP \quad (5)$$

$$FPR = \frac{FP}{FP + TN} FPR = FP + TNFP \quad (6)$$

Confusion Matrix

A confusion matrix is a tabular illustration of the TPTP, FPFP, TNTN, and FNFN counts. It is possible to make these types of plots for datasets with any number of classes present. Below is an example of the 2-class confusion matrix:

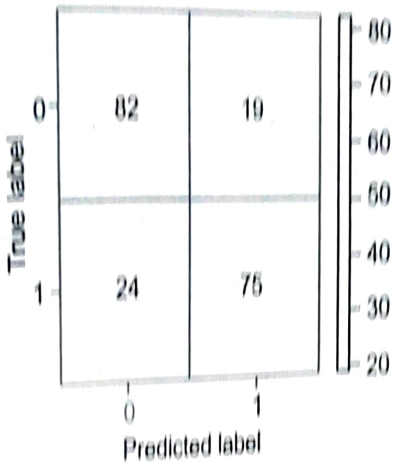


Fig. 5.10.

True classes are indicated on the left vertical. Predicted classes are on the bottom horizontal. A perfect classifier would have all counts on the diagonal, and zeros everywhere else.

Python Coding Examples

In this section we will work in Python to demonstrate how to measure performance of a classification model. First let's import the necessary packages:

```
### imports ###
import numpy as np
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_curve, plot_confusion_matrix
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

Next let's use the make_classification function to generate some data:

```
### create dataset ###
```

```
X,y = make_classification(n_samples=1000,
                          n_features=20,
                          n_informative=5,
                          n_redundant=2,
                          n_classes=2,
                          random_state=42)
```

A total of 1000 samples are generated, with 20 features in the matrix \bold{X}X. Of these features, only 5 are informative, and therefore useful for modelling. Another 2 features are redundant, meaning these are linear combinations of the informative features. The vector of labels yy contains 2 classes.Our next step is to separate the data into train and test sets. Therefore, we can use the train_test_split function:

```
## do a train-test split ##
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Twenty percent of the data is allocated for testing. The remaining 80% will be used to train our model. Our model here is a decision tree classifier, which does not require feature scaling. Therefore, let's proceed to train the model:

```
## declare and fit a classifier ##
tree = DecisionTreeClassifier()
tree.fit(X_train,y_train)
```

Now we can generate predictions with our trained classifier on the test dataset. Furthermore, we can evaluate the performance of the model using metrics (1), (2), (3), and (4). We will make use of the functionality available through scikit-learn:

```
## obtain predictions & measure performance ##
y_pred = tree.predict(X_test)
acc  = accuracy_score(y_test,y_pred)
pre  = precision_score(y_test,y_pred)
rec  = recall_score(y_test,y_pred)
f1s  = f1_score(y_test,y_pred)
print("Accuracy score: %.2f" % acc)
```

```
print("Precision score: %.2f" % pre)
print("Recall score: %.2f" % rec)
print("F1 score: %.2f" % f1s)
Accuracy score: 0.92
Precision score: 0.95
Recall score: 0.88
F1 score: 0.91
```

The classifier is functioning well, considering we performed no hyperparameter ning. The decision tree does a very good job at correctly identifying the '1' classes recision), at the expense of finding all of them (recall). The overall score ccuracy/f1) is quite reasonable.

Finally, we can visualise our results by making a confusion matrix and ROC plots:

```
## produce a confusion matrix ##
plot_confusion_matrix(tree, X_test, y_test)
plt.show()
```

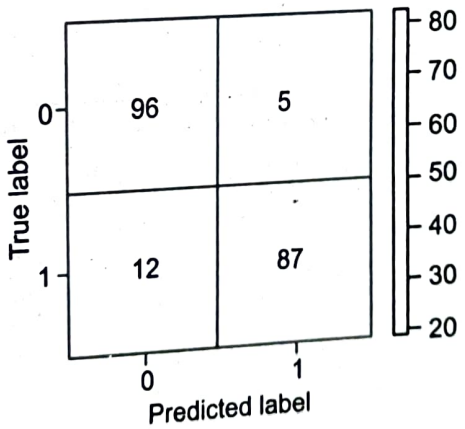


Fig. 5.11.

```
## produce a ROC plot ##
#obtain prediction probabilities
y_prob = tree.predict_proba(X_test)
#calculate false & true positive rates
fpr,tpr,_ = roc_curve(y_test, y_prob[:,1])
#construct plot
```

```
plt.plot(fpr,tpr)
plt.title('Receiver Operating Characteristic')
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".8"), plt.plot([1, 1] , c=".8")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

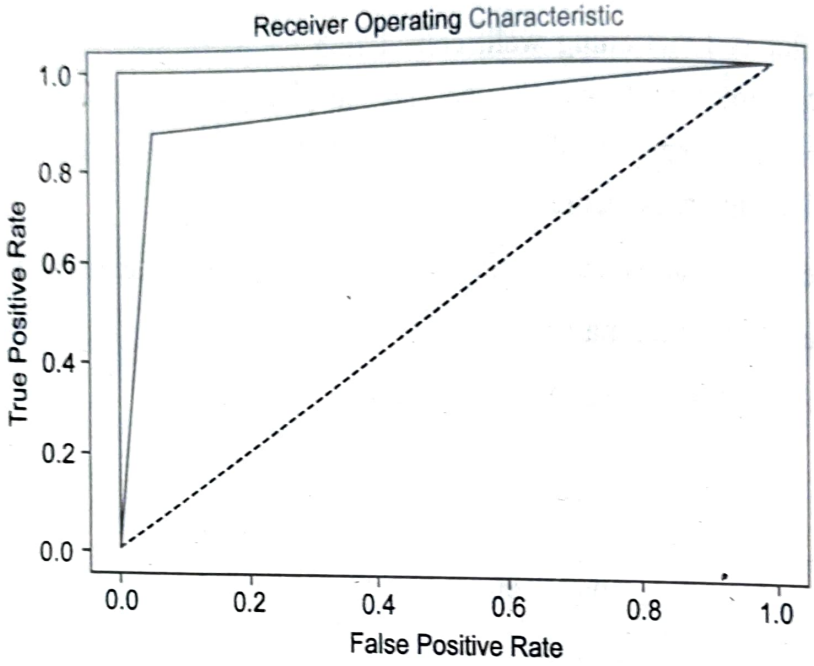


Fig. 5.12.

These plots demonstrate reasonably good results, consistent with the scores we calculated earlier. Next steps could include the following two approaches. First, we could try out different classification models. Second, we could perform hyperparameter tuning to optimise any given model.

5.8. ASSESSING A SINGLE CLASSIFICATION ALGORITHM AND TWO CLASSIFICATION ALGORITHMS – T TEST

What is the Classification Algorithm?

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then

ssifies new observation into a number of classes or groups. Such as, Yes or No, 0 , Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or tegories.

Unlike regression, the output variable of Classification is a category, not a value, ch as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a pervised learning technique, hence it takes labeled input data, which means it ntains input with the corresponding output. In classification algorithm, a discrete tput function(y) is mapped to input variable(x).

$y = f(x)$, where y = categorical output

The best example of an ML classification algorithm is Email Spam Detector. The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.

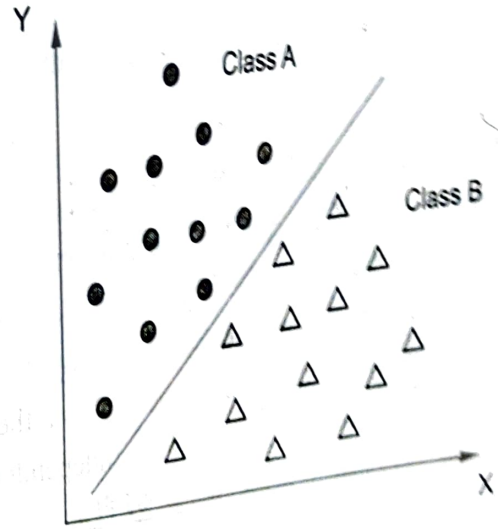


Fig. 5.13.

The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

Binary Classifier: If the classification problem has only two possible outcomes, then it is called as Binary Classifier.

Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

Multi-class Classifier: If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.

Example: Classifications of types of crops, Classification of types of music.

Learners in Classification Problems:

In the classification problems, there are two types of learners:

Lazy Learners: Lazy Learner firstly stores the training dataset and wait until it receives the test dataset. In Lazy learner case, classification is done on the basis of the most related data stored in the training dataset. It takes less time in training but more time for predictions.

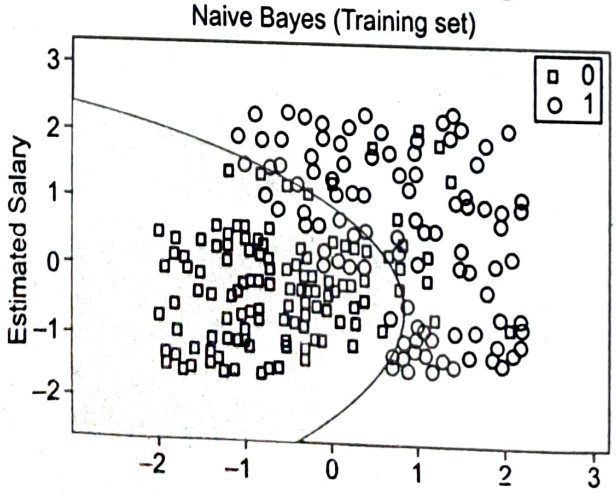
Example: K-NN algorithm, Case-based reasoning

Eager Learners: Eager Learners develop a classification model based on a training dataset before receiving a test dataset. Opposite to Lazy learners, Eager Learner takes more time in learning, and less time in prediction. Example: Decision Trees, Naïve Bayes, ANN.

Popular algorithms that can be used for binary classification include:

5.8.1. NAIVE BAYES

The Naive Bayes method is a supervised learning algorithm based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.



5.8.2. LOGISTIC REGRESSION

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.Logistic Regression is a Machine Learning algorithm used to make predictions to find the value of a dependent variable such as the condition of a tumor (malignant or benign), classification of email (spam or not spam), or admission into a university (admitted or not admitted) by learning from independent variables (various features relevant to the problem).

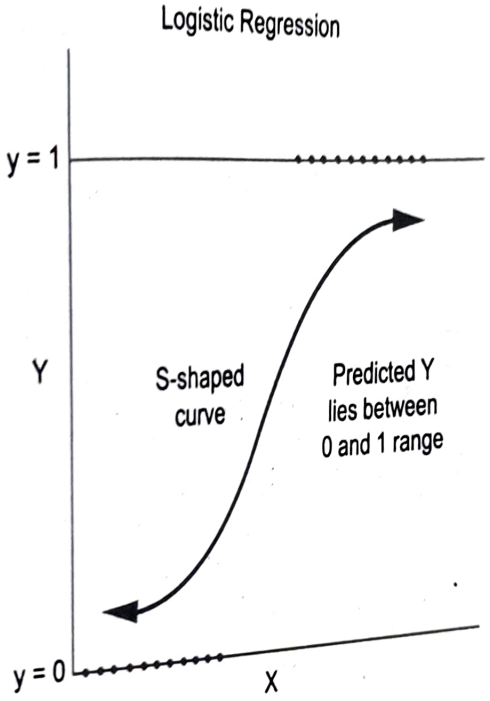


Fig. 5.14.

5.8.3. K-NEAREST NEIGHBOURS

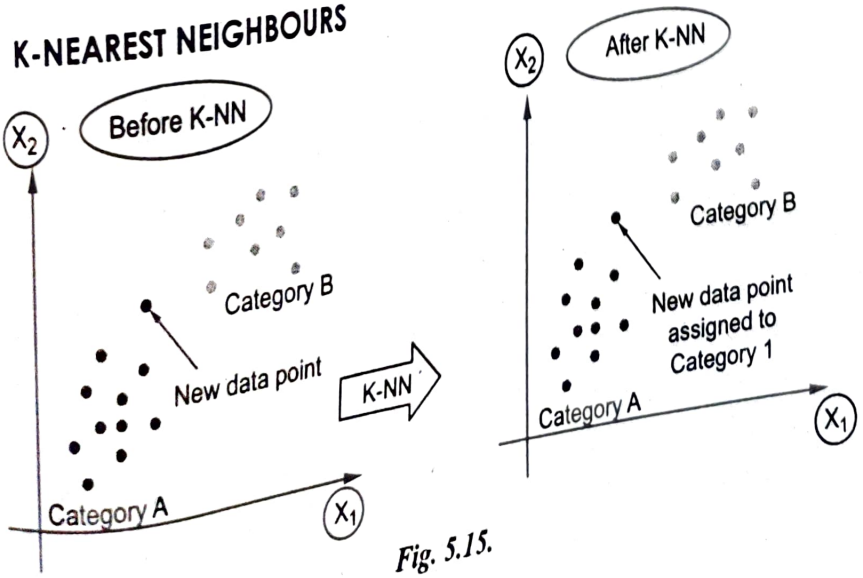


Fig. 5.15.

K-Nearest Neighbour (K-NN) algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

5.8.4. SUPPORT VECTOR MACHINE

Support vector machine is based on statistical approaches. Her we try to find a hyperplane that best separates the two classes. SVM finding the maximum margin between the hyperplanes that means maximum distances between the two classes. SVM works best when the dataset is small and complex. When the data is perfectly linearly separable only then we can use Linear SVM. When the data is not linearly separable then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a linear approach.

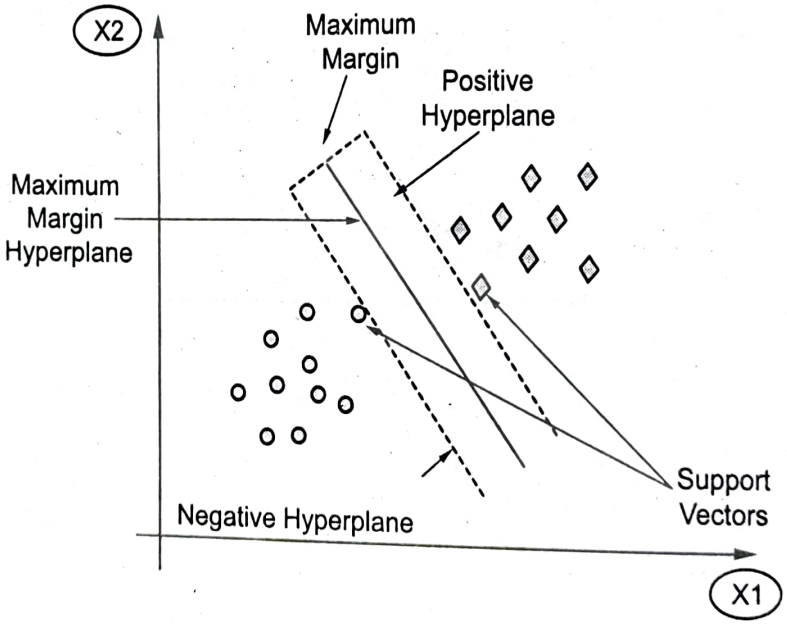


Fig. 5.16.

5.8.5. DECISION TREE

A Decision Tree is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules (if-else) inferred from the data features. Decision trees can perform both classification and regression tasks, so you'll see authors refer to them as CART algorithm: Classification and Regression

Tree. This is an umbrella term, applicable to all tree-based algorithms, not just decision trees.

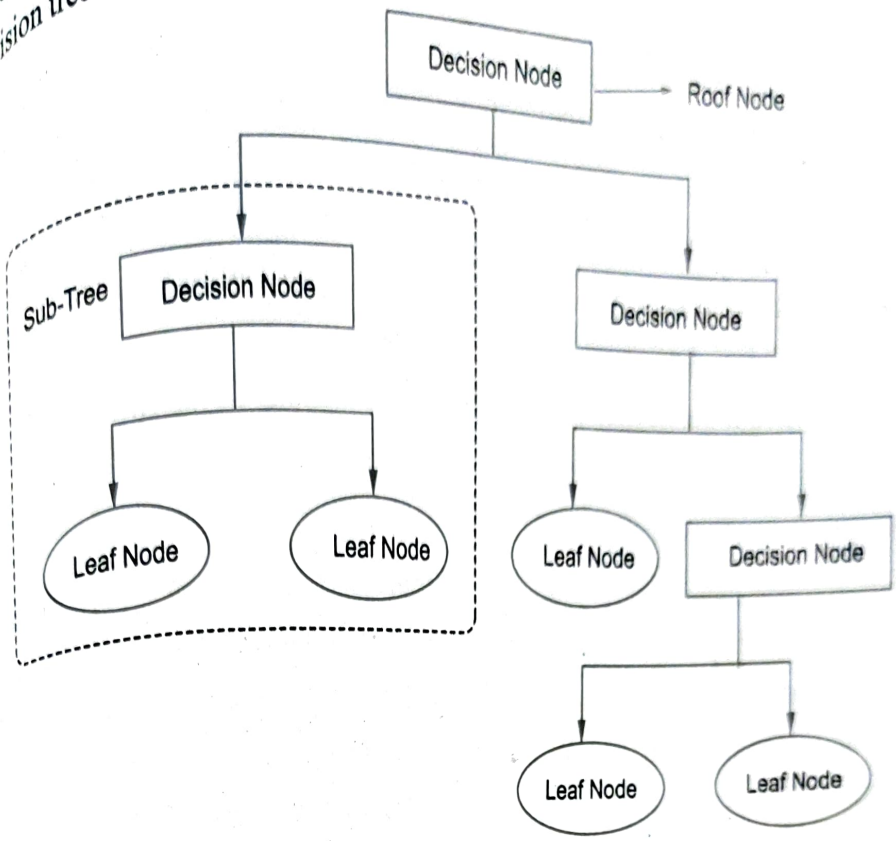


Fig. 5.17.

5.9. T-TEST

A T-test is the final statistical measure for determining differences between two means that may or may not be related. The testing uses randomly selected samples from the two categories or groups. It is a statistical method in which samples are chosen randomly, and there is no perfect normal distribution.

The type of T-test to be conducted is decided by whether the samples to be analyzed are from the same category or distinct categories. The inference obtained in the process indicates the probability of the mean differences to have happened by chance. The test is useful when comparing population age, length of crops from two different species, student grades, etc.

5.9.1. T-TEST EXPLAINED

A T-test studies a set of data gathered from two similar or different groups to determine the probability of the difference in the result than what is usually obtained.

The accuracy of the test depends on various factors, including the distribution patterns used and the variants influencing the collected samples.

Depending on the parameters, the test is conducted, and a T-value is obtained as the statistical inference of the probability of the usual resultant being driven by chance. For example, if one wishes to figure out if the mean of the length of petals of a flower belonging to two different species is the same, a T-test can be done. The user can select petals randomly from two other species of that flower and come to a standard conclusion.

The final T-test interpretation could be obtained in either of the two ways: A null hypothesis signifies that the difference between the means is zero and where both the means are shown as equal.

An alternate hypothesis implies the difference between the means is different from zero. This hypothesis rejects the null hypothesis, indicating that the data set is quite accurate and not by chance. This T-test, however, is only valid and should be done when the mean or average of only two categories or groups needs to be compared. As soon as the number of comparisons to be made is more than two, conducting this is not recommended.

Performing a *t* test

The *t* test estimates the true difference between two group means using the ratio of the difference in group means over the pooled standard error of both groups. You can calculate it manually using a formula, or use statistical analysis software.

T test formula

The formula for the two-sample *t* test (a.k.a. the Student's t-test) is shown below.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(s^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right))}}$$

In this formula, t is the t value, x_1 and x_2 $A = \pi r^2$ the means of the two groups being compared, s^2 is the pooled standard error of the two groups, and n_1 and n_2 are the number of observations in each of the groups. A larger t value shows that the difference between group means is greater than the pooled standard error, indicating a more significant difference between the groups.

You can compare your calculated *t* value against the values in a critical value chart (e.g., Student's *t* table) to determine whether your *t* value is greater than what would be expected by chance. If so, you can reject the null hypothesis and conclude that the two groups are in fact different.

Interpreting test results

If you perform the *t* test for your flower hypothesis in R, you will receive the following output:

```
Welch Two Sample t-test

data:  Petal.Length by Species
t = -33.719, df = 30.196, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.331287 -3.836713
sample estimates:
   mean in group setosa mean in group virginica
                  1.456                  5.540
```

The output provides:

1. An explanation of what is being compared, called **data** in the output table.
2. The ***t* value**: -33.719. Note that it's negative; this is fine! In most cases, we only care about the absolute value of the difference, or the distance from 0. It doesn't matter which direction.
3. The **degrees of freedom**: 30.196. Degrees of freedom is related to your sample size, and shows how many 'free' data points are available in your test for making comparisons. The greater the degrees of freedom, the better your statistical test will work.
4. The ***p* value**: 2.2e-16 (i.e. 2.2 with 15 zeros in front). This describes the probability that you would see a *t* value as large as this one by chance.
5. A statement of the **alternative hypothesis** (*Ha*). In this test, the *Ha* is that the difference is not 0.
6. The 95% **confidence interval**. This is the range of numbers within which the true difference in means will be 95% of the time. This can be changed from 95% if you want a larger or smaller interval, but 95% is very commonly used.

7. The **mean** petal length for each group.

t test example

From the output table, we can see that the difference in means for our sample data is −4.084 (1.456 − 5.540), and the confidence interval shows that the true difference in means is between −3.836 and −4.331. So, 95% of the time, the true difference in means will be different from 0. Our *p* value of 2.2e−16 is much smaller than 0.05, so we can **reject the null hypothesis** of no difference and say with a high degree of confidence that the **true difference in means is not equal to zero**.

5.10. MCNEMAR' S TEST

This is a non-parametric test for the paired nominal data. This test is used when we want to find the change in proportion for the paired data. This test is also known as McNemar's Chi-Square test. This is because the test statistic has a chi-square distribution.

Assumptions for the McNemar Test:

Below are the main assumptions for the test:

We must have one nominal variable with two categories (dichotomous variables) and one independent variable with two connected groups. Two sets of the dependent variables must be mutually exclusive. In simple words, participants cannot be part of more than one group. The sample must be a random sample. Mc Nemer test is utilized for two related examples as a part of circumstances where the states of mind of individuals are noted previously, then after the fact treatment to test the essentialness of progress in sentiment if any.

The Mc Nemer test is especially helpful when the information speaks the truth two related samples. For the most part this information is utilized as a part of circumstances where the states of mind of individuals are noted before overseeing the treatment and are then contrasted and investigations in the wake of managing the treatment. It can along these lines be said that utilizing McNemer test we can judge if there is any adjustment in the demeanors or supposition of individuals subsequent to regulating the treatment with the utilization of table as demonstrated as follows:

Do not favour

Before Treatment	After Treatment	
		Favour
Favour	A	B
Do not favour	C	D

As can be seen C and B don't change their supposition and show 'Do Not Favour' and 'Favour' individually even after the treatment has been administered .However, A which was good before treatment demonstrates a 'Do Not Favour' reaction after treatment and vice versa for D. It can hence be said that A + DA + D shows change in individuals' reaction. The null hypothesis for Mc Nemer test is that $(A + D)^2(A + D)^2$ cases change in one direction and the same proportion of change takes place in other direction.

McNemer test statistic uses a transformed _test model as follows:

$$x_2 = (|A - D| - 1)^2(A + D) \times 2 = (|A - D| - 1)^2(A + D)$$

Acceptance Criteria: If the calculated value is less then the table value, accept null hypothesis.

Rejection Criteria: If the calculated value is more than table value then null hypothesis is rejected.

Illustration

In a before and after experiment the responses obtained from 300 respondents were classified as follows:

Do not favour

Before Treatment	After Treatment	
		Favour
Favour	60 = A	90 = B
Do not favour	120 = C	30 = D

Test at 5% significance level, using McNemer test if there is any significant difference in the opinion of people after the treatment.

Solution:

There is no difference in the opinion of people even after the experiment.

The test statistic is calculated using the formula:

$$x_2 = (|A-D| - 1)^2(A + D) = (|60 - 30| - 1)^2(60 + 30) = 9.34 \times 2$$

$$= (|A - D| - 1)^2(A + D) = (|60 - 30| - 1)^2(60 + 30) = 9.34$$

The value of test at 5% significance level for 1 D.F. is 3.84. Since the test is greater than the table value, the null hypothesis is rejected i.e. the opinion of people has changed after the treatment.

5.11. K-FOLD CV PAIRED T TEST

K-fold cross-validated paired t-test procedure is a common method for comparing the performance of two models (classifiers or regressors) and addresses some of the drawbacks of the resampled t-test procedure; however, this method has still the problem that the training sets overlap and is not recommended to be used in practice.

To explain how this method works, let's consider to estimator (e.g., classifiers) A and B. Further, we have a labeled dataset D. In the common hold-out method, we typically split the dataset into 2 parts: a training and a test set. In the k-fold cross-validated paired t-test procedure, we split the test set into k parts of equal size, and each of these parts is then used for testing while the remaining $k - 1$ parts (joined together) are used for training a classifier or regresson.

In each k-fold cross-validation iteration, we then compute the difference in performance between A and B in each so that we obtain k difference measures. Now, by making the assumption that these k differences were independently drawn and follow an approximately normal distribution, we can compute the following t statistic with $k - 1$ degrees of freedom according to Student's t test, under the null hypothesis that the models A and B have equal performance:

$$t = \frac{\bar{p}\sqrt{k}}{\sqrt{\sum_{i=1}^{k} (p^{(i)} - \bar{p})^2 / (k-1)}}$$

Here, $p(i)$ computes the difference between the model performances in the i^{th} iteration, $p(i) = p(i)A - p(i)B$, $\bar{p}$ represents the average difference between the classifier performances, $\bar{p} = 1k \sum k\, i = 1\, p(i)$.

Once we computed the t statistic we can compute the p value and compare it to our chosen significance level, e.g., $\alpha = 0.05$. If the p value is smaller than α, we reject the null hypothesis and accept that there is a significant difference in the two models.

The problem with this method, and the reason why it is not recommended to be used in practice, is that it violates an assumption of Student's t test.

Example 1 - K-fold cross-validated paired t test

Assume we want to compare two classification algorithms, logistic regression and a decision tree

Algorithm

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from mlxtend.data import iris_data
from sklearn.model_selection import train_test_split
X, y = iris_data()
clf1 = LogisticRegression(random_state=1)
clf2 = DecisionTreeClassifier(random_state=1)
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.25,
                     random_state=123)
score1 = clf1.fit(X_train, y_train).score(X_test, y_test)
score2 = clf2.fit(X_train, y_train).score(X_test, y_test)
print('Logistic regression accuracy: %.2f%%' % (score1*100))
print('Decision tree accuracy: %.2f%%' % (score2*100))
```

Logistic regression accuracy: 97.37%

Decision tree accuracy: 94.74%

Note that these accuracy values are not used in the paired t-test procedure as new test/train splits are generated during the resampling procedure, the values above are just serving the purpose of intuition. Now, let's assume a significance threshold of α

= 0.05 α = 0.05 for rejecting the null hypothesis that both algorithms perform equally well on the dataset and conduct the k-fold cross-validated t-test:

```
from mlxtend.evaluate import paired_ttest_kfold_cv
t, p = paired_ttest_kfold_cv(estimator1=clf1,
                             estimator2=clf2,
                             X=X, y=y,
                             random_seed=1)

print('t statistic: %.3f' % t)
print('p value: %.3f' % p)
```

t statistic: -1.861

p value: 0.096

Since p > αp > α, we cannot reject the null hypothesis and may conclude that the performance of the two algorithms is not significantly different. While it is generally not recommended to apply statistical tests multiple times without correction for multiple hypothesis testing, let us take a look at an example where the decision tree algorithm is limited to producing a very simple decision boundary that would result in a relatively bad performance:

```
clf2 = DecisionTreeClassifier(random_state=1, max_depth=1)

score2 = clf2.fit(X_train, y_train).score(X_test, y_test)
print('Decision tree accuracy: %.2f%%' % (score2*100))

t, p = paired_ttest_kfold_cv(estimator1=clf1,
                             estimator2=clf2,
                             X=X, y=y,
                             random_seed=1)

print('t statistic: %.3f' % t)
print('p value: %.3f' % p)
```

Decision tree accuracy: 63.16%
statistic: 13.491
p value: 0.000
Assuming that we conducted this test also with a significance level of $\alpha = 0.05$ $\alpha =$ 05, we can reject the null-hypothesis that both models perform equally well on this taset, since the p-value ($p < 0.001$ $p < 0.001$) is smaller than $\alpha\alpha$.
The difference between the model performances ($p(i) = p(i)A - p(i)Bp(i) = pA(i)$ $pB(i)$) are not normal distributed because $p(i)ApA(i)$ and $p(i)Bp$ $B(i)$ are not dependent the $p(i)p(i)$'s themselves are not independent because training sets verlap.

TWO MARKS QUESTIONS AND ANSWERS (PART-A)

1. ***What are the guidelines for machine learning experiments***

Machine learning projects are highly iterative; as you progress through the ML lifecycle, you'll find yourself iterating on a section until reaching a satisfactory level of performance, then proceeding forward to the next task (which may be circling back to an even earlier step). Moreover, a project isn't complete after you ship the first version; you get feedback from real-world interactions and redefine the goals for the next iteration of deployment.

2. ***State the Model exploration***

- Establish baselines for model performance
- Start with a simple model using initial data pipeline
- Overfit simple model to training data
- Stay nimble and try many parallel (isolated) ideas during early stages
- Find SoTA model for your problem domain (if available) and reproduce results, then apply to your dataset as a second baseline

3. ***List the Properties of an experiment, trial and trial component:***

- An Experiment is uniquely characterized by its objective or hypothesis
- An Experiment usually contains more than one Trial, one Trial for each variable set.

- A Trial is uniquely characterized by its variable set, sampled from the variable space defined by you.
- A Trial component is any artifact, parameter or job that is associated with a specific Trial.
- A Trial component is usually part of a Trial, but it can exist independent of an experiment or trial.
- A Trial component cannot be directly associated with an Experiment. It has to be associated with a Trial which is associated with an Experiment.

4. ***What is the Validation Set Approach?***

Validation Set approach is very simple method and frequently used method when there is sufficiently enough amount of observations to get reasonable results. It is basically dividing the data that we have to two place as train set and validation set (or holdout set), and building model on train set , then checking the model accuracy on validation set. And resulting accuracy from validation set is the estimate about the real test data(unseen data).

5. ***Define LOOCV.***

Leave One Out Cross Validation method is addressed to the drawbacks of the validation set approach and it is also simple method as validation set approach. Main point here is to take each observation as a validation set one time. It means that if we have "n" number of observations, we will fit model "n" times. And in every try we will keep one observation as a test sample, and will train the model on "$n - 1$" observations.

6. ***List down the types of resampling methods:***

- Bootstrap Sampling
- K Fold Cross Validation
- Leave One Out Cross Validation

7. ***State the Resampling Method.***

Resampling methods are very useful and beneficial in statistics and machine learning to fit more accurate models, model selection and parameter tuning. They draw samples from train data and fit model to check the variability of model and get additional information. We cannot best sure of the result of the model by just unique fit without testing on different sample or samples. It can be

computationally expensive because of fitting model more than one, but recent improvements tackle this issue easily without too much effort.

What is a Cross Validation – Kfold?

8. Cross validation resamples without replacement and thus produces surrogate data sets that are smaller than the original. These data sets are produced in a systematic way so that after a pre-specified number k of surrogate data sets, each of the n original cases has been left out exactly once. This is called k-fold cross validation or leave-x-out cross validation with $x = n / k$, e.g. leave-one-out cross validation omits 1 case for each surrogate set, i.e. $k = n$.

What do you mean by Bootstrapping?

9. Bootstrapping is a resampling method that is used in machine learning. It is a widespread technique due to its flexibility since it does not require anything other than your training dataset. Bootstrap Sampling comes from the ideas around just the Boostrap. The Bootstrap is a flexible and powerful statistical tool that brings us closer to our sample's true population parameters.

What is The Difference Between Bootstrapping And Cross-Validation?

10. Bootstrapping and Cross-Validation are both sampling methods of statistical inference.

In general, statistical inference uses data from a sample to make estimates or predictions about a population. Both bootstrapping and cross-validation are used to estimate the performance of our population's "standard error" or, more simply, how our machine-learning algorithm will do in a production system on unseen data.

The main difference between the two methods is that bootstrapping is a resampling technique, while cross-validation is a partitioning technique. Bootstrapping involves random sampling with replacement from the training data set to create multiple new training sets.

This means that bootstrapping will lower the variance for our machine-learning model.

What is the Classification Algorithm?

11. The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In

Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories.

12. *Define Naive Bayes*

The Naive Bayes method is a supervised learning algorithm based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

13. *Write about Support Vector Machine*

Support vector machine is based on statistical approaches. Here we try to find a hyperplane that best separates the two classes. SVM finding the maximum margin between the hyperplanes that means maximum distances between the two classes. SVM works best when the dataset is small and complex. When the data is perfectly linearly separable only then we can use Linear SVM. When the data is not linearly separable then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a linear approach.

14. *What is a T-test*

A T-test is the final statistical measure for determining differences between two means that may or may not be related. The testing uses randomly selected samples from the two categories or groups. It is a statistical method in which samples are chosen randomly, and there is no perfect normal distribution.

15. *Define McNemar's test*

McNemar's Test: This is a non-parametric test for the paired nominal data. This test is used when we want to find the change in proportion for the paired data. This test is also known as McNemar's Chi-Square test. This is because the test statistic has a chi-square distribution.

16. ***What is a K-fold CV paired t test?***

K-fold cross-validated paired t-test procedure is a common method for comparing the performance of two models (classifiers or regressors) and addresses some of the drawbacks of the resampled t-test procedure; however, this method has still the problem that the training sets overlap and is not recommended to be used in practice.

PART-B & C

1. Discuss about the guidelines for machine learning experiments
2. Explain Project lifecycle in machine learning
3. Discuss on the term Cross Validation
4. Compare and contrast types of resampling methods
5. Demonstrate on Cross Validation
6. How will you Evaluate a ML model using K-Fold CV?
7. What is Bootstrapping? How do you implement it in python?
8. Broadly explain on measuring classifier performance.
9. Briefly explain the Popular algorithms that can be used for binary classification
10. Explain the method of Performing a t test.
11. Demonstrate the McNemar Test.

****