SWAMINARAYAN COLLEGE OF ENNG. & TECH. (DEGREE)

MACHINE LEARNING
(3170724)

PREPARED BY:- Porwal Harsh
201150107005

SUBMITTED TO:-
COMPUTER DEPARTMENT
SCET, SAIJ, KALOL

SWAMINARAYAN COLLEGE OF ENNG. & TECH. (DEGREE)

CERTIFICATE

This Certificate is awarded to Mr. Porwal Harsh from **SWAMINARAYAN COLLEGE OF ENNG. & TECH. (DEGREE)** having enrollment no:- 201150107005 has completed practical work of "MACHINE LEARNING" during the 7th semester on term 2023-24.

INTERNAL GUIDE **DATE OF SUBMISSION** **H.O.D**

Prof. Nimesh Vaidya **Prof. Nimesh Vaidya**

INDEX

Subject: Machine Learning
Subject Code: 3170724

Branch: Computer Engineering
Semester: 7th

Sr. No.	Title	Page No.	Date	Sign
1	Introduction to Machine Learning, tools, and libraries.			
2	Introduction to Data Preprocessing.			
3	Implement Simple Linear Regression on the given dataset.			
4	Implement Support Vector Machine on given dataset.			
5	Implement K-means algorithm on the given dataset.			
6	Implement K-fold cross validation on the given dataset.			
7	Implement PCA algorithm on the given dataset.			
8	Implement Apriori algorithm on the given dataset.			
9	Implement Thompson Sampling algorithm on the given dataset.			
10	Design Implement and Train an artificial neural network on the given dataset.			

FACULTY: Smit Nayak

H.O.D: Prof.Nimesh Vaidya

Practical – 1

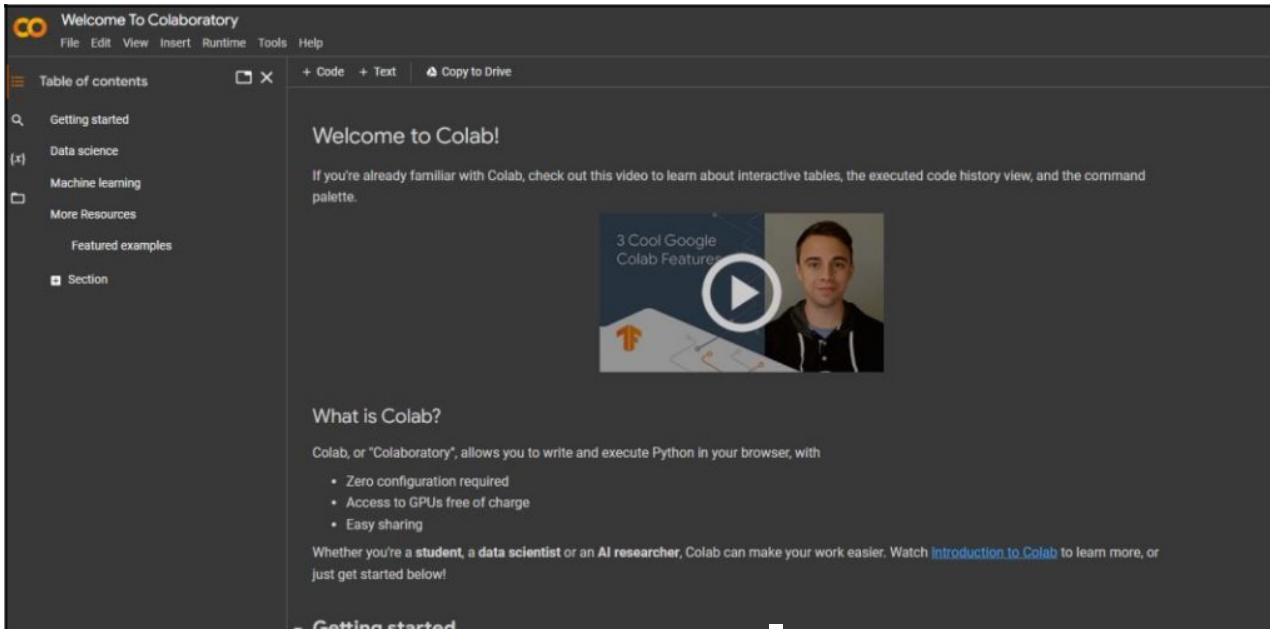
Machine Learning Tools and Libraries

Aim: Introduction to Machine Learning, tools, and libraries.

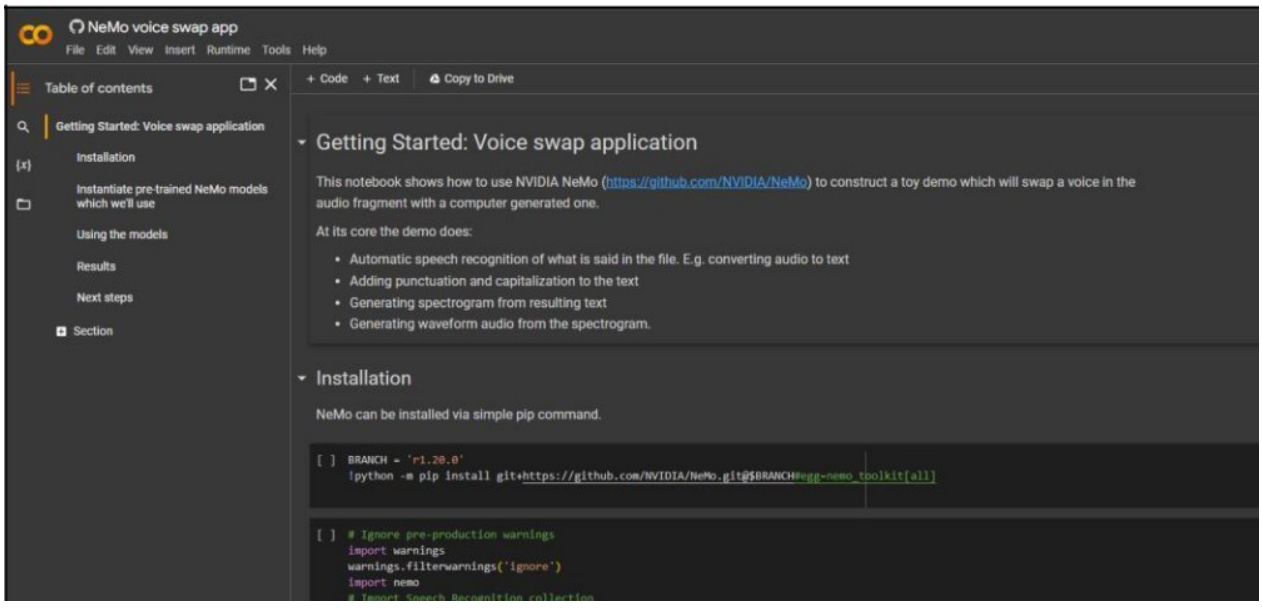
Exercise:

1. Explore Google Collaboratory and other IDE tools.

Answer

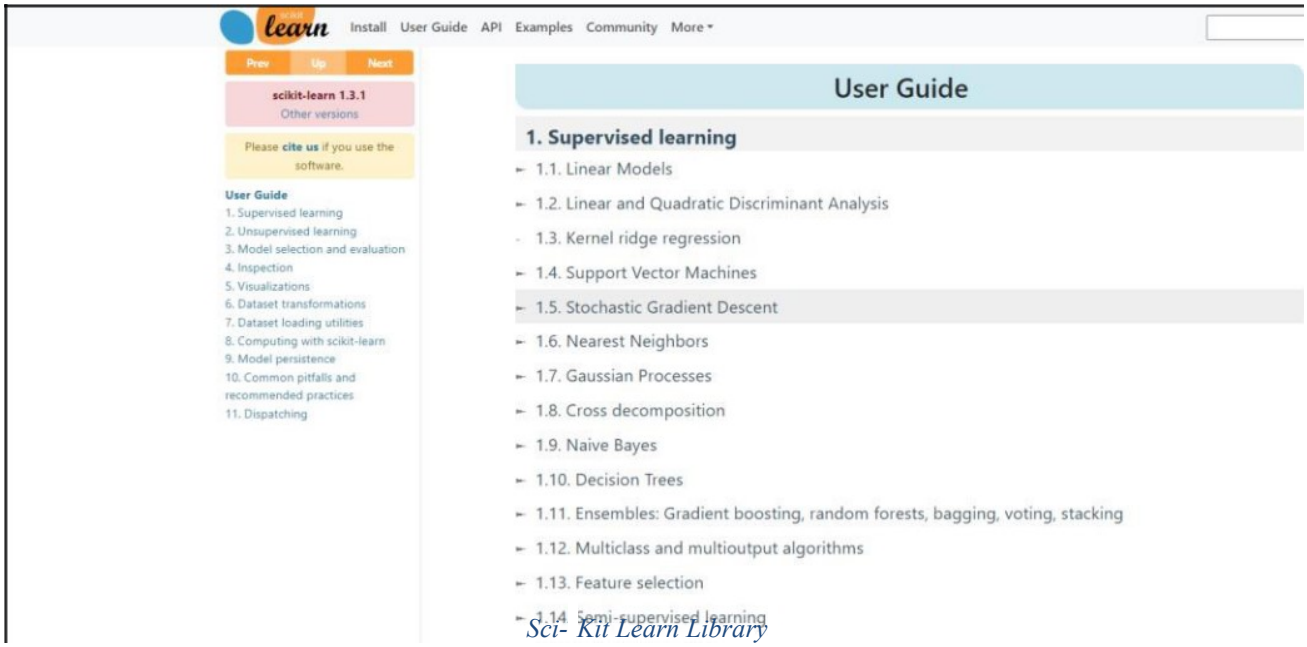


Google Collaboratory

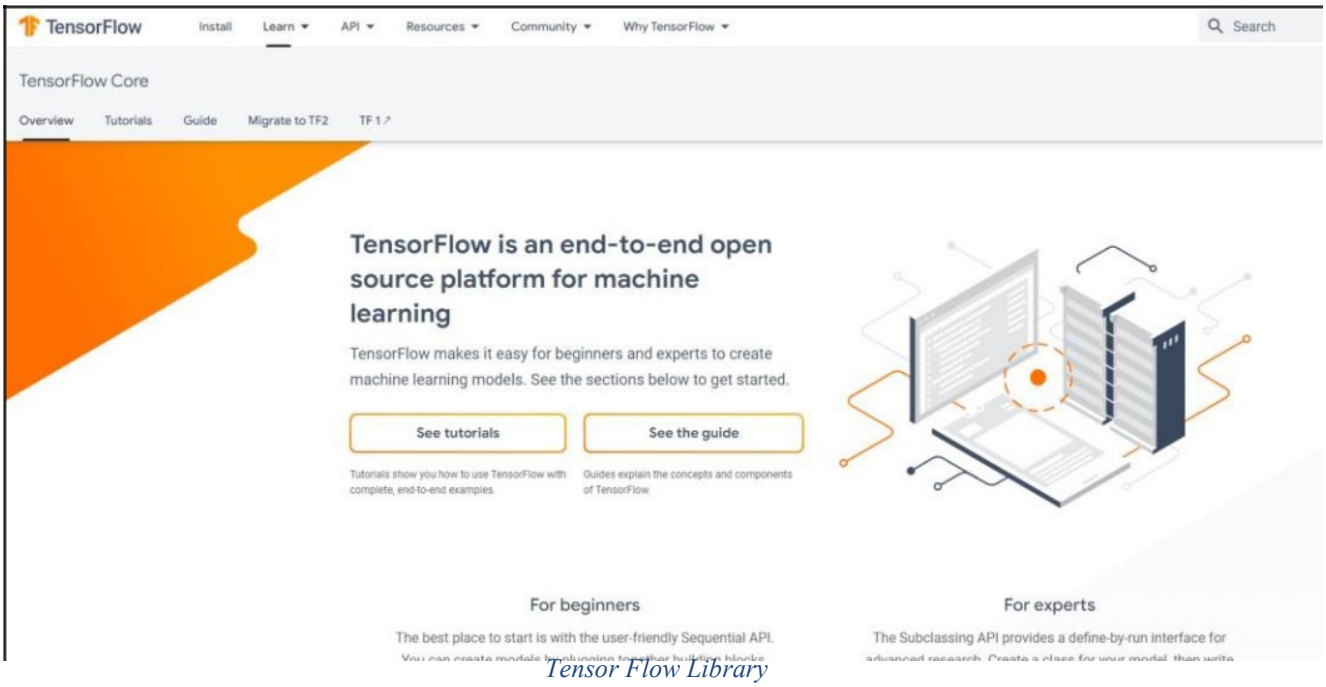


2. Explore the Documentation of various libraries of machine learning.

Answer



Sci- Kit Learn Library



Tensor Flow Library

Practical – 2

Data Preprocessing

Aim: Introduction to Data Preprocessing.

Exercise:

1. On the given dataset perform following data preprocessing techniques:
   i. Handle missing values.
   ii. Encoding of Categorical variables.
   iii. Converting Data of different range to same range iv.
   Splitting dataset into training set and test set.

Answer:

Step 1:

Importing the libraries

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [14]:
```
dataset = pd.read_csv('Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

In [15]:
```
print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

Importing the Libraries and Dataset

Step 2:

Taking care of missing data

In [17]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
```

In [18]:
```
print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 63777.77777777778]
 ['France' 35.0 58000.0]
```

Handling Missing Values

Step 3:

Encoding categorical data

Encoding the Independent Variable

In [19]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainde
X = np.array(ct.fit_transform(X))
```

In [20]:
```
print(X)
```

```
[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.77777777777778 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

Encoding the Dependent Variable

In [21]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

Encoding of Categorical Variables

Step 4:

Feature Scaling

In [28]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [29]:
```
print(X_train)
```

```
[[-0.77459667 -0.57735027  1.29099445 -0.19159184 -1.07812594]
 [-0.77459667  1.73205081 -0.77459667 -0.01411729 -0.07013168]
 [ 1.29099445 -0.57735027 -0.77459667  0.56670851  0.63356243]
 [-0.77459667 -0.57735027  1.29099445 -0.30453019 -0.30786617]
 [-0.77459667 -0.57735027  1.29099445 -1.90180114 -1.42046362]
 [ 1.29099445 -0.57735027 -0.77459667  1.14753431  1.23265336]
 [-0.77459667  1.73205081 -0.77459667  1.43794721  1.57499104]
 [ 1.29099445 -0.57735027 -0.77459667 -0.74014954 -0.56461943]]
```

In [30]: print(X_test)

Converting Data of different range into same range

Step 5:

```
In [23]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rand

In [24]: print(X_train)

[[0.0 0.0 1.0 38.77777777777778 52000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 35.0 58000.0]]

In [25]: print(X_test)

[[0.0 1.0 0.0 30.0 54000.0]
 [1.0 0.0 0.0 37.0 67000.0]]

In [26]: print(y_train)

[0 1 0 0 1 1 0 1]
```

Splitting Dataset into training set and test set

Practical – 3

Simple Linear Regression Aim:

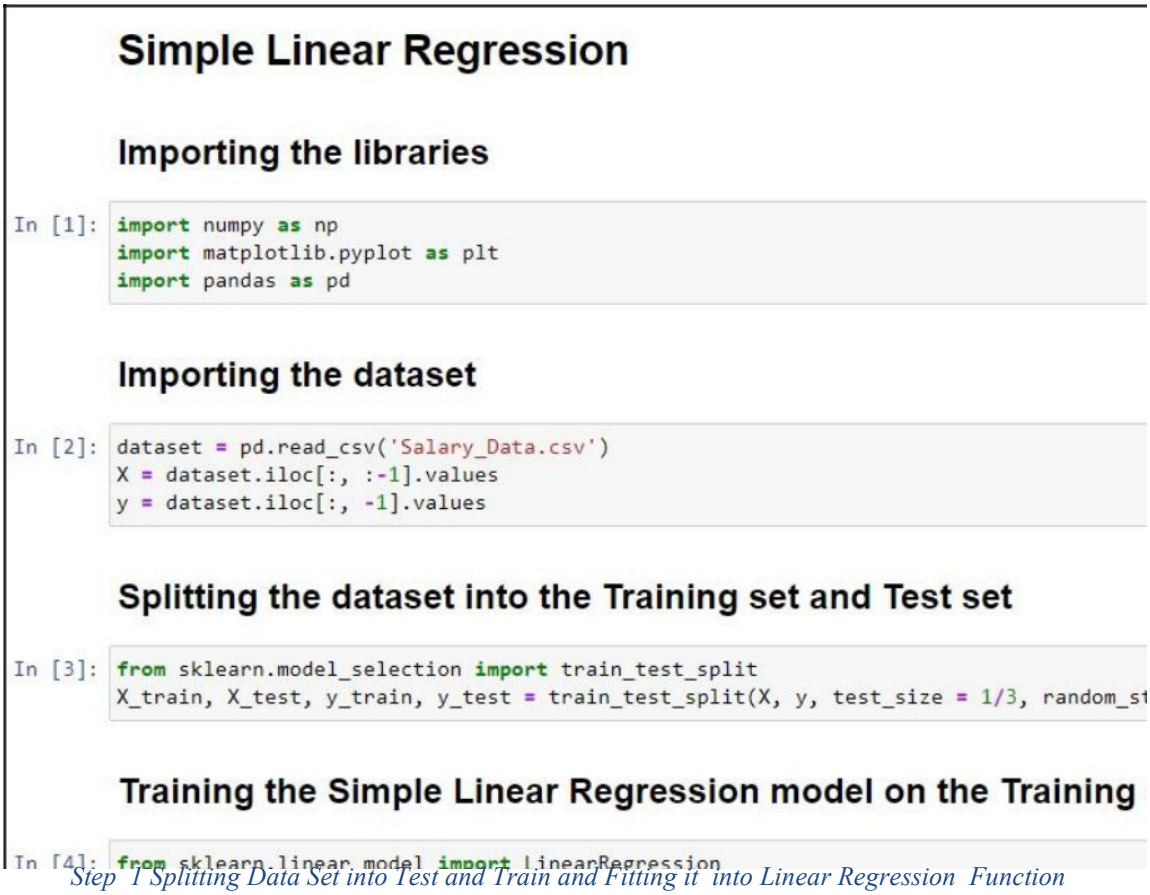
To Study Simple Linear Regression

Exercise:

1. Implement Simple Linear Regression on the given dataset.

Answer:

Step 1:



Simple Linear Regression

Importing the libraries

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

Importing the dataset

```
In [2]: dataset = pd.read_csv('Salary_Data.csv')
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
In [3]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_st
```

Training the Simple Linear Regression model on the Training

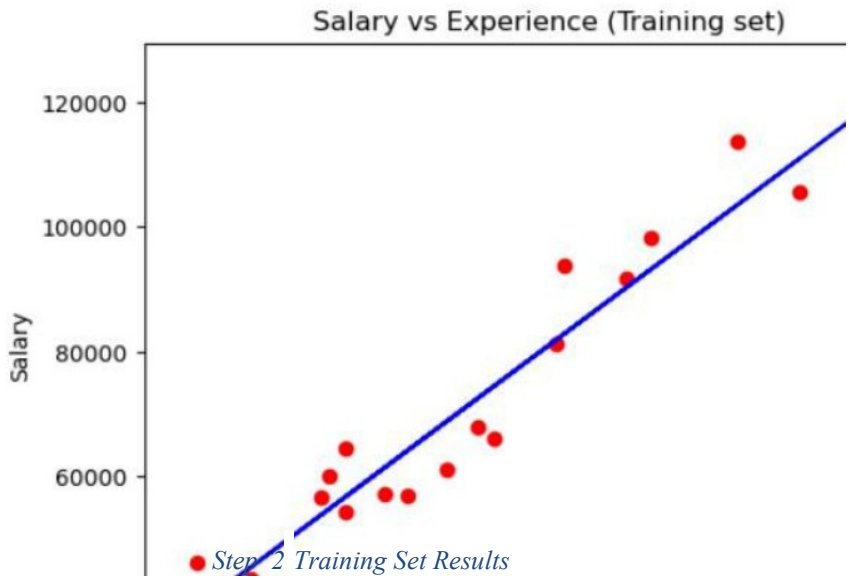
```
In [4]: from sklearn.linear_model import LinearRegression
```

Step 1 Splitting Data Set into Test and Train and Fitting it into Linear Regression Function

Step 2:

Visualising the Training set results

```
In [6]: plt.scatter(X_train, y_train, color = 'red')
        plt.plot(X_train, regressor.predict(X_train), color = 'blue')
        plt.title('Salary vs Experience (Training set)')
        plt.xlabel('Years of Experience')
        plt.ylabel('Salary')
        plt.show()
```

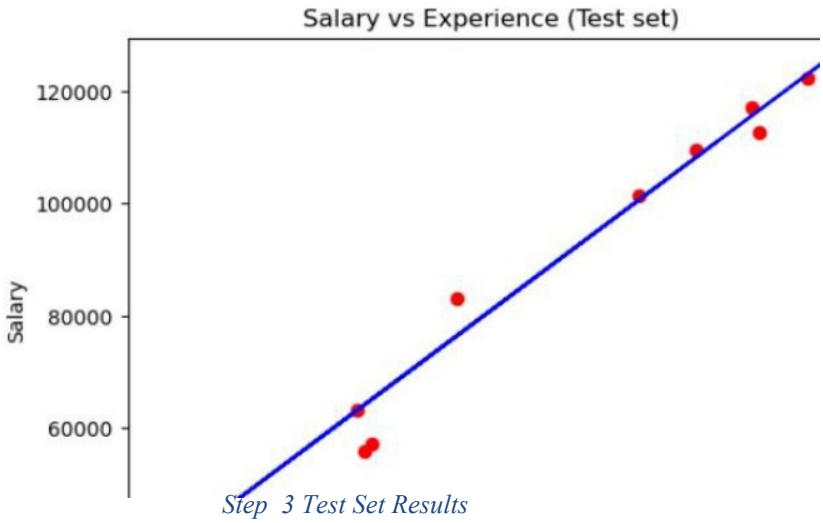


Step 2 Training Set Results

Step 3:

Visualising the Test set results

```
In [7]: plt.scatter(X_test, y_test, color = 'red')
        plt.plot(X_train, regressor.predict(X_train), color = 'blue')
        plt.title('Salary vs Experience (Test set)')
        plt.xlabel('Years of Experience')
        plt.ylabel('Salary')
        plt.show()
```



Step 3 Test Set Results

Practical – 4

Support Vector Machine (SVM)

Aim: To study Support Vector Machine.

Exercise:

1. Implement Support Vector Machine on given dataset.

Answer:

Step 1:

Support Vector Machine (SVM)

Importing the libraries

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

Importing the dataset

```
In [2]: dataset = pd.read_csv('Social_Network_Ads.csv')
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

Importing t he Data set and Splitting into test and train

Step 2:

Training the SVM model on the Training set

```
In [11]: from sklearn.svm import SVC
         classifier = SVC(kernel = 'linear', random_state = 0)
         classifier.fit(X_train, y_train)
```

```
Out[11]: SVC(kernel='linear', random_state=0)
```

Predicting a new result

```
In [12]: print(classifier.predict(sc.transform([[30,87000]])))
```

```
[0]
```

Predicting the Test set results

```
In [13]: y_pred = classifier.predict(X_test)
         print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

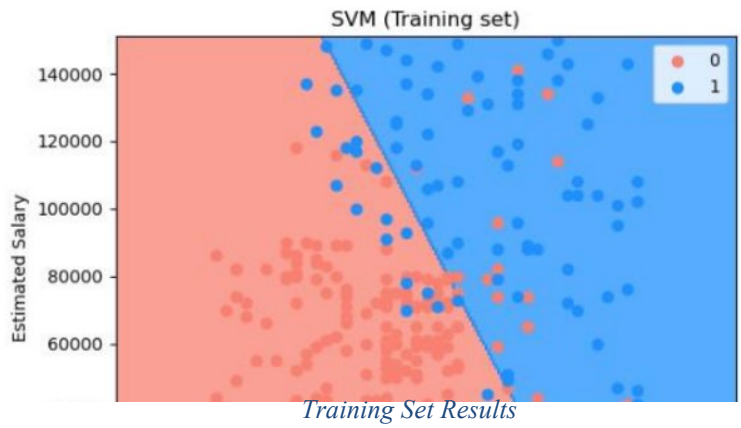
```
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
```

Training SVM Model on Dataset and Predicting test results.

Step 3:

Visualising the Training set results

```
In [16]: from matplotlib.colors import ListedColormap
         X_set, y_set = sc.inverse_transform(X_train), y_train
         X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, st
                              np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000
         plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resh
                      alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
         plt.xlim(X1.min(), X1.max())
         plt.ylim(X2.min(), X2.max())
         for i, j in enumerate(np.unique(y_set)):
             plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodge
         plt.title('SVM (Training set)')
         plt.xlabel('Age')
         plt.ylabel('Estimated Salary')
         plt.legend()
         plt.show()
```

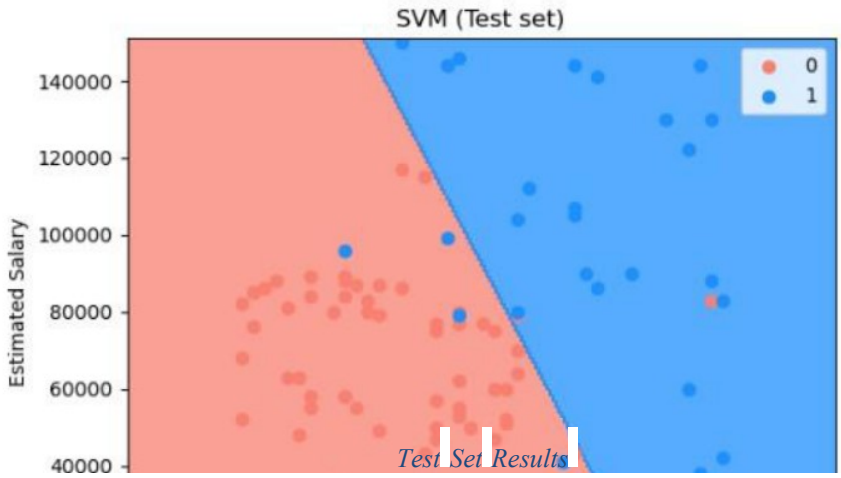


Training Set Results

Step 4:

Visualising the Test set results

In [17]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0.
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.
             alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue')
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Test Set Results

5
K- Means Algorithm Aim:

To study how to implement K-means Algorithm

Exercise:

1. Implement K-means algorithm on the given dataset.

Answer:

Step 1:

Importing the libraries

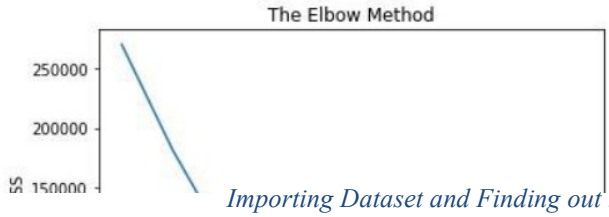
```
In [0]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

Importing the dataset

```
In [0]: dataset = pd.read_csv('Mall_Customers.csv')
        X = dataset.iloc[:, [3, 4]].values
```

Using the elbow method to find the optimal number of clusters

```
In [3]: from sklearn.cluster import KMeans
        wcss = []
        for i in range(1, 11):
            kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
            kmeans.fit(X)
            wcss.append(kmeans.inertia_)
        plt.plot(range(1, 11), wcss)
        plt.title('The Elbow Method')
        plt.xlabel('Number of clusters')
        plt.ylabel('WCSS')
        plt.show()
```



Importing Dataset and Finding out number of clusters

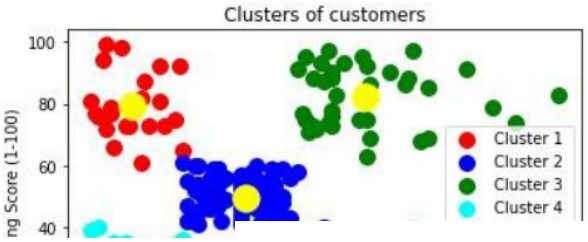
Step 2:

Training the K-Means model on the dataset

```
In [0]: kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
        y_kmeans = kmeans.fit_predict(X)
```

Visualising the clusters

```
In [5]: plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
        plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
        plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
        plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
        plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
        plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label
        plt.title('Clusters of customers')
        plt.xlabel('Annual Income (k$)')
        plt.ylabel('Spending Score (1-100)')
        plt.legend()
        plt.show()
```



Training K Means Model and Visualizing the Clusters

K-fold Cross Validation

Aim: To study how to do implementation of K-fold cross validation.

Exercise:

1. Implement K-fold cross validation on the given dataset.

Answer:

Step 1:

```
In [1]: from sklearn.datasets import load_breast_cancer
        import pandas as pd
        from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import KFold
        from sklearn.metrics import accuracy_score

In [2]: data = load_breast_cancer(as_frame = True)
        df = data.frame

In [3]: X = df.iloc[:,:-1]
        Y = df.iloc[:,-1]

In [4]: #Implementing k-fold cross-validation
        k = 5
        k_fold = KFold(n_splits = k, random_state = None)
        Lr = LogisticRegression(solver = 'liblinear')

In [5]: acc_scores = []

In [6]: # Looping over each split to get the accuracy score of each split
        for training_index, testing_index in k_fold.split(X):
            X_train, X_test = X.iloc[training_index,:], X.iloc[testing_index
            Y_train, Y_test = Y.iloc[training_index] , Y.iloc[testing_index]
```

Implementing K-Fold Cross Validation using Logistic Regression Model

Step 2:

```
In [10]: # Predicting values for the testing dataset
         Y_pred = Lr.predict(X_test)

In [11]: # Calculatinf accuracy score using in-built sklearn accuracy_scor
         acc = accuracy_score(Y_pred , Y_test)
         acc_scores.append(acc)

In [12]: # Calculating mean accuracy score
         mean_acc_score = sum(acc_scores) / k

In [13]: print("Accuracy score of each fold: ", acc_scores)
         print("Mean accuracy score: ", mean_acc_score)
```

Predicting Values and Calculating Accuracy

7

Dimensionality Reduction Aim:

To study Introduction to dimensionality reduction.

Exercise:

1. Implement PCA algorithm on the given dataset.

Answer:

Step 1:

Importing the libraries

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import warnings
        warnings.filterwarnings('ignore')
```

Importing the dataset

```
In [2]: dataset = pd.read_csv('Wine.csv')
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
In [3]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
```

Feature Scaling

```
In [4]: from sklearn.preprocessing import StandardScaler
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
```

Importing Dataset and Applying PCA

Step 2:

Training the Logistic Regression model on the Training s

```
In [6]: from sklearn.linear_model import LogisticRegression
        classifier = LogisticRegression(random_state = 0)
        classifier.fit(X_train, y_train)
```

```
Out[6]: LogisticRegression
        LogisticRegression(random_state=0)
```

Making the Confusion Matrix ¶

```
In [7]: from sklearn.metrics import confusion_matrix, accuracy_score
        y_pred = classifier.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        print(cm)
        accuracy_score(y_test, y_pred)
```

```
[[14  0  0]
 [ 1 15  0]
```

Training Logistic Regression Model

Step 3:

Visualising the Training set results

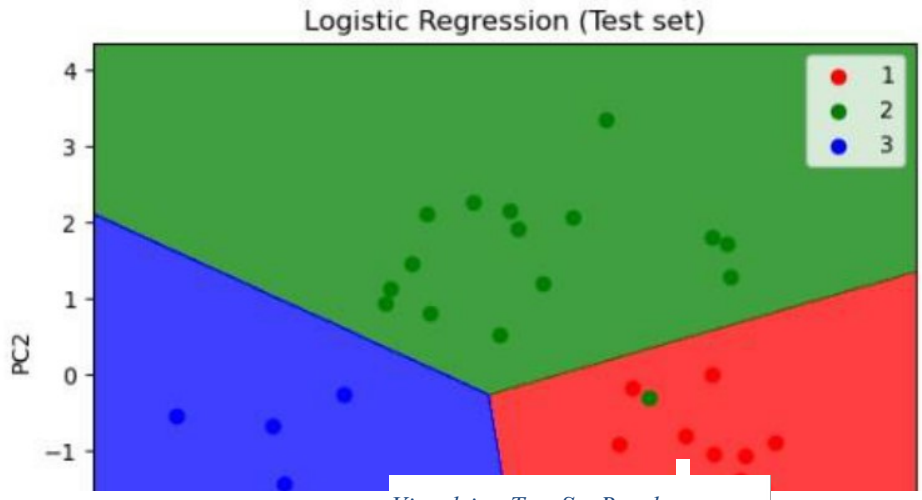
In [8]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```

Visualizing Training S et Results

Step 4:

Visualising the Test set results

```
In [9]: from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.show()
```



Visualsing Test Set Results

Practical – 8

Association Rule Learning Aim:

To study introduction to Association Rule Learning.

Exercise:

1. Implement Apriori algorithm on the given dataset.

Answer:

Step 1:

Apriori

Importing the libraries

```
In [1]: !pip install apyori
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py): started
  Building wheel for apyori (setup.py): finished with status 'done'
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5975 sha256=33b1730bb5b7519779
ac0e131b53cfc1025945a
  Stored in directory: c:\users\dharm\appdata\local\pip\cache\wheels\84\45\a4\8ade6576f75410d8162c6da1
3
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

Data Preprocessing

```
In [3]: dataset = pd.read_csv('Market_Basket_Optimisation.csv', header = None)
        transactions = []
        for i in range(0, 7501):
```

Importing Libraries and Training Apriori Model

Step 2:

Visualising the results

Displaying the first results coming directly from the output of the apriori function

```
In [5]: results = list(rules)

In [6]: results

Out[6]: [RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered_stat
(items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.290598290598290
5)]),
 RelationRecord(items=frozenset({'mushroom cream sauce', 'escalope'}), support=0.005732568990801226, or
dStatistic(items_base=frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), confiden
ft=3.790832696715049)]),
 RelationRecord(items=frozenset({'pasta', 'escalope'}), support=0.005865884548726837, ordered_statistic
s_base=frozenset({'pasta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, lift=4.
 RelationRecord(items=frozenset({'honey', 'fromage blanc'}), support=0.003332888948140248, ordered_stat
(items_base=frozenset({'fromage blanc'}), items_add=frozenset({'honey'}), confidence=0.245098039215686
9)]),
 RelationRecord(items=frozenset({'herb & pepper', 'ground beef'}), support=0.015997866951073192, ordere
tistic(items_base=frozenset({'herb & pepper'}), items_add=frozenset({'ground beef'}), confidence=0.3234
9938411349285)]),
 RelationRecord(items=frozenset({'tomato sauce', 'ground beef'}), support=0.005332622317024397, ordered
istic(items_base=frozenset({'tomato sauce'}), items_add=frozenset({'ground beef'}), confidence=0.377358
9481324083)]),
 RelationRecord(items=frozenset({'light cream', 'olive oil'}), support=0.003199573390214638, ordered_st
ic(items_base=frozenset({'light cream'}), items_add=frozenset({'olive oil'}), confidence=0.205128205128
519573)]),
 RelationRecord(items=frozenset({'whole wheat pasta', 'olive oil'}), support=0.007998933475536596, orde
tatistic(items_base=frozenset({'whole wheat pasta'}), items_add=frozenset({'olive oil'}), confidence=0.
4.122410097642296)]),
 RelationRecord(items=frozenset({'pasta', 'shrimp'}), support=0.005065991201173177, ordered_statistics=
base=frozenset({'pasta'}), items_add=frozenset({'shrimp'}), confidence=0.3220338983050847, lift=4.5066
```

Putting the results well organised into a Pandas DataFrame

Visualizing Results and Putting Results into Pandas Data Frame

Displaying the results non sorted

```
In [8]: resultsinDataFrame
```

Out[8]:

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	light cream	chicken	0.004533	0.290598	4.843951
1	mushroom cream sauce	escalope	0.005733	0.300699	3.790833
2	pasta	escalope	0.005866	0.372881	4.700812
3	fromage blanc	honey	0.003333	0.245098	5.164271
4	herb & pepper	ground beef	0.015998	0.323450	3.291994
5	tomato sauce	ground beef	0.005333	0.377358	3.840659
6	light cream	olive oil	0.003200	0.205128	3.114710
7	whole wheat pasta	olive oil	0.007999	0.271493	4.122410
8	pasta	shrimp	0.005066	0.322034	4.506672

Displaying the results sorted by descending lifts

```
In [9]: resultsinDataFrame.nlargest(n = 10, columns = 'Lift')
```

Out[9]:

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
3	fromage blanc	honey	0.003333	0.245098	5.164271
0	light cream	chicken	0.004533	0.290598	4.843951
2	pasta	escalope	0.005866	0.372881	4.700812
8	pasta	shrimp	0.005066	0.322034	4.506672

Displaying Sorted and Non -Sorted Results

Practical – 9
Reinforcement Learning

Aim: To study the Introduction of Reinforcement Learning

Exercise:

1. Implement Thompson Sampling algorithm on the given dataset.

Answer:

Step 1:

Thompson Sampling

Importing the libraries

In [0]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [0]:
```
dataset = pd.read_csv('Ads_CTR_Optimisation.csv')
```

Implementing Thompson Sampling

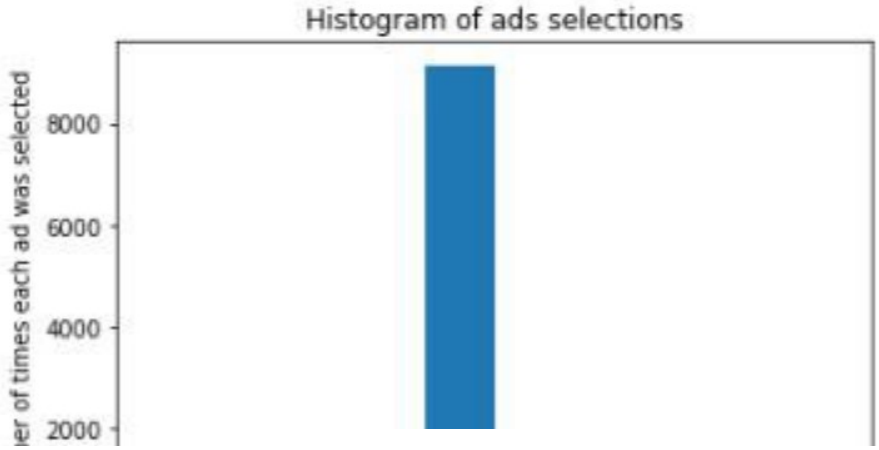
In [0]:
```
import random
N = 10000
d = 10
ads_selected = []
numbers_of_rewards_1 = [0] * d
numbers_of_rewards_0 = [0] * d
total_reward = 0
for n in range(0, N):
    ad = 0
    max_random = 0
    for i in range(0, d):
        random_beta = random.betavariate(numbers_of_rewards_1[i] + 1, numbers_of_rewards_0[i] + 1)
        if random_beta > max_random:
```

Implementing Thompson Sampling

Step 2:

Visualising the results - Histogram

```
In [4]: plt.hist(ads_selected)
        plt.title('Histogram of ads selections')
        plt.xlabel('Ads')
        plt.ylabel('Number of times each ad was selected')
        plt.show()
```



Visualizing the Results Using Histogram

Practical – 10

Deep Learning

Aim: To Study the introduction of Deep Learning.

Exercise:

1. Design Implement and Train an artificial neural network on the given dataset.

Answer:

Step 1: Data Preprocessing

Artificial Neural Network

Importing the libraries

```
In [0]: import numpy as np
        import pandas as pd
        import tensorflow as tf
```

```
In [2]: tf.__version__
Out[2]: '2.2.0'
```

Part 1 - Data Preprocessing

Importing the dataset

```
In [0]: dataset = pd.read_csv('Churn_Modelling.csv')
        X = dataset.iloc[:, 3:-1].values
        y = dataset.iloc[:, -1].values
```

```
In [4]: print(X)
[[619 'France' 'Female' ... 1 1 101348.88]
 [608 'Spain' 'Female' ... 0 1 112542.58]
 [502 'France' 'Female' ... 1 0 113931.57]
 ...
```

Importing Libraries and Dataset

Encoding categorical data

Label Encoding the "Gender" column

```
In [0]: from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
        X[:, 2] = le.fit_transform(X[:, 2])
```

```
In [7]: print(X)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

One Hot Encoding the "Geography" column

```
In [0]: from sklearn.compose import ColumnTransformer
        from sklearn.preprocessing import OneHotEncoder
        ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
        X = np.array(ct.fit_transform(X))
```

```
In [9]: print(X)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
```

Encoding Categorical Data

Splitting the dataset into the Training set and Test set

```
In [0]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
```

Feature Scaling

```
In [0]: from sklearn.preprocessing import StandardScaler
        sc = StandardScaler()
```

Splitting into Train and Test and Scaling Data

Step 2: Building the ANN

Part 2 - Building the ANN

Initializing the ANN

```
In [0]: ann = tf.keras.models.Sequential()
```

Adding the input layer and the first hidden layer

```
In [0]: ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the second hidden layer

```
In [0]: ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the output layer

Building ANN

Step 3: Training the ANN

Compiling the ANN

```
In [0]: ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accu
```

Training the ANN on the Training set

```
In [17]: ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
Epoch 1/100
250/250 [==============================] - 0s 1ms/step - loss: 0.8037 - accura
Epoch 2/100
250/250 [==============================] - 0s 1ms/step - loss: 0.5291 - accura
Epoch 3/100
250/250 [==============================] - 0s 1ms/step - loss: 0.4888 - accura
Epoch 4/100
250/250 [==============================] - 0s 1ms/step - loss: 0.4668 - accura
Epoch 5/100
250/250 [==============================] - 0s 1ms/step - loss: 0.4478 - accura
Epoch 6/100
250/250 [==============================] - 0s 1ms/step - loss: 0.4302 - accura
Epoch 7/100
250/250 [==============================] - 0s 1ms/step - loss: 0.4123 - accura
Epoch 8/100
```

Comping And Train ing ANN

Step 4: Making Predictions and Evaluating the Model

Part 4 - Making the predictions and evaluating the model

Predicting the result of a single observation

```
Use our ANN model to predict if the customer with the following informations will

Geography: France

Credit Score: 600

Gender: Male

Age: 40 years old

Tenure: 3 years

Balance: \$ 60000

Number of Products: 2

Does this customer have a credit card ? Yes

Is this customer an Active Member: Yes

Estimated Salary: \$ 50000

So, should we say goodbye to that customer ?
```

Predicting Using Sample Data

Predicting the Test set results

In [19]:
```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y
```

```
[[0 0]
 [0 1]
 [0 0]
 ...
 [0 0]
 [0 0]
 [0 0]]
```

Making the Confusion Matrix

In [20]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Predicting Test Results and Making Confusion Matrix