Assignment # 04

Data Mining

Submitted to : Dr Muzamil

Submitted by: Abdullah Aslam(Team Lead) 01-134172-005

Ahsan Goheer 01-134172-008

Section : BSCS – 7A

Date : January 5, 2021

TABLE OF CONTENTS

INTRODUCTION

This dataset is associated with the world-famous titanic incident that took place back in April 1912. This dataset is available at Kaggle as an open competition [1] in the form of 2 files (train and test) for Kagglers to apply machine learning to predict the survival of a person.

The train.csv file has 12 columns and 891 rows while the test.csv file has 11 columns and 418 rows.

OVERVIEW OF THE DATA

FIRST FIVE ROWS OF THE TRAIN.CSV DATA

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

FIRST FIVE ROWS OF THE TEST.CSV DATA

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

FINDINGS BASED ON THE DATA

- Out of 891 Passengers in the data, 342 survived and 549 did not survive.
- Out of 891 Passengers in the data, 577 are males and 314 are females.
- Number of females who survived: 233
- Number of males who survived: 109
- Passengers in class 1 had a higher chance of survival, then followed by class 2 and then class 3.
- Passengers with at least one parent or child had a higher chance of survival.
- Passengers with 1 or 2 siblings or spouse had a higher chance of survival.
- Passengers embarked from 'Cherboug' had the most survivors, followed by 'South Hampton' and then 'Queenstown'.
- Null Values in the data are as follows:

Cabin	**687**
Age	**177**
Embarked	**2**

- Survival rate based on the Title of the Passengers:

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Others	0.347826

Based on the data 79% of the married women, 70% of bachelorettes, 57% of bachelors and 34% of people with other titles survived the titanic disaster.

- Outliers in the dataset are as follows:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.00	C23 C25 C27	S
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.00	C23 C25 C27	S
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8	2	CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8	2	CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr. Frederick	male	NaN	8	2	CA. 2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.55	NaN	S
341	342	1	1	Fortune, Miss. Alice Elizabeth	female	24.0	3	2	19950	263.00	C23 C25 C27	S
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8	2	CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.55	NaN	S

COMPARISON OF DIFFERENT PREDICTION MODELS

CODE

IMPORTING THE MODULES

```
import numpy as np
import pandas as pd
from sklearn.metrics import plot_confusion_matrix
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.impute import SimpleImputer
from sklearn.naive_bayes import GaussianNB
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from keras.layers import Dense,Dropout
from keras.models import Sequential
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
```

IMPORTING DATA FROM CSV

```
train_dataset = pd.read_csv('./train.csv')
test_dataset = pd.read_csv('./test.csv')
```

REMOVING OUTLIERS FROM THE DATA

```
def outliers(df,features):
    indices = []
    for f in features:
        Q1 = np.percentile(df[f],25)
        Q3 = np.percentile(df[f],75)
        IQR = Q3 - Q1
        outlier_step = IQR * 1.5
        outlier_list_col = df[(df[f] < Q1 - outlier_step) | (df[f] > Q3 + outlier_step)].index
        indices.extend(outlier_list_col)
    indices = Counter(indices)
    outliers = list(i for i, v in indices.items() if v > 2)
    return outliers
```

```
train_dataset = train_dataset.drop(outliers(train_dataset,["Age","SibSp","Parch","Fare"]),axis =
0).reset_index(drop = True)
```

SPLITTING INDEPENDENT AND DEPENDENT VARIABLES

```
X_train = train_dataset.iloc[:, [2,4,5,6,7,9,11]].values
y_train = train_dataset.iloc[:, 1].values
```

CONVERTING TEST DATA INTO NUMPY ARRAY

```
X_test = test_dataset.iloc[:, [1,3,4,5,6,8,10]].values
```

CONVERTING GENDER FROM CATEGORICAL TO BINARY VARIABLE

```
label_encoder_gender = LabelEncoder()
X_train[:, 1] = label_encoder_gender.fit_transform(X_train[:, 1])
X_test[:, 1] = label_encoder_gender.transform(X_test[:, 1])
```

FILLING MISSING VALUES OF EMBARKED WITH MODE

```
most_frequent_embarked = max(dict(train_dataset.Embarked.value_counts()))
# for training data
filling_indices = [x for x in range(len(X_train)) if X_train[x, -1] != 'S' and X_train[x, -1] != 'Q' and
X_train[x, -1] != 'C']
X_train[filling_indices, -1] = most_frequent_embarked

# for testing data
filling_indices = [x for x in range(len(X_test)) if X_test[x, -1] != 'S' and X_test[x, -1] != 'Q' and
X_test[x, -1] != 'C']
X_test[filling_indices, -1] = most_frequent_embarked
```

FILLING MISSING AGE VALUES WITH MEAN AGE

```
imputer_age = SimpleImputer(strategy='mean')
X_train[:, [2]] = imputer_age.fit_transform(X_train[:, [2]])
X_test[:, [2]] = imputer_age.transform(X_test[:, [2]])
```

FILLING MISSING FARE VALUES WITH MEAN

```
imputer_fare = SimpleImputer(strategy='mean')
X_train[:, [5]] = imputer_fare.fit_transform(X_train[:, [5]])
X_test[:, [5]] = imputer_fare.transform(X_test[:, [5]])
```

ONEHOT ENCODING PASSENGER CLASS

```
ct_pclass = ColumnTransformer([('one_hot_encoder', OneHotEncoder(categories='auto'),
[0])],remainder='passthrough')
X_train = ct_pclass.fit_transform(X_train)
```

SKIPPING DUMMY VARIABLE TRAP

```
ct_pclass = ColumnTransformer([('one_hot_encoder', OneHotEncoder(categories='auto'),
[0])],remainder='passthrough')
X_train = ct_pclass.fit_transform(X_train)
```

CONVERTING EMBARKED LOCATION TO SPARSE MATRIX

```
embarked_encoder = LabelEncoder()
X_train[:, -1] = embarked_encoder.fit_transform(X_train[:, -1])
X_test[:, -1] = embarked_encoder.transform(X_test[:, -1])
```

APPLYING Z SCORE NORMALIZATION TO AGE

```
sc_age = StandardScaler()
X_train[:, [5]] = sc_age.fit_transform(X_train[:, [5]])
X_test[:, [5]] = sc_age.transform(X_test[:, [5]])
```

APPLYING Z SCORE NORMALIZATION TO FARE

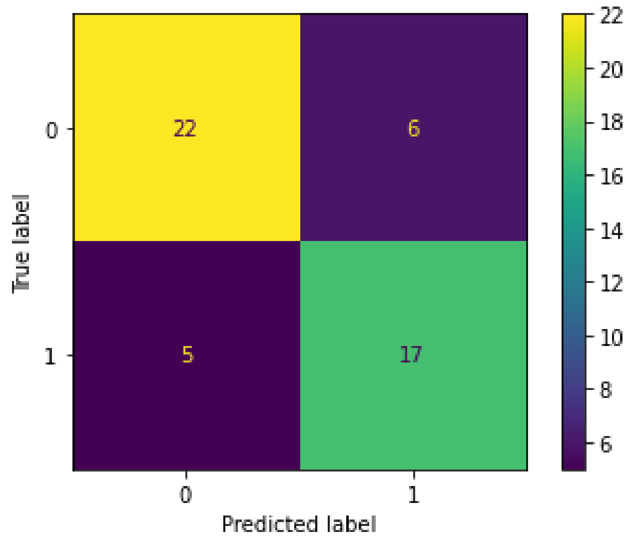
```
sc_fare = StandardScaler()
X_train[:, [-1]] = sc_fare.fit_transform(X_train[:, [-1]])
X_test[:, [-1]] = sc_fare.transform(X_test[:, [-1]])
```

APPLYING PCA FOR FEATURE EXTRACTION

```
pca = PCA(n_components=8)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

NAÏVE BAYES CLASSIFER USING GAUSSIAN NAÏVE BAYES

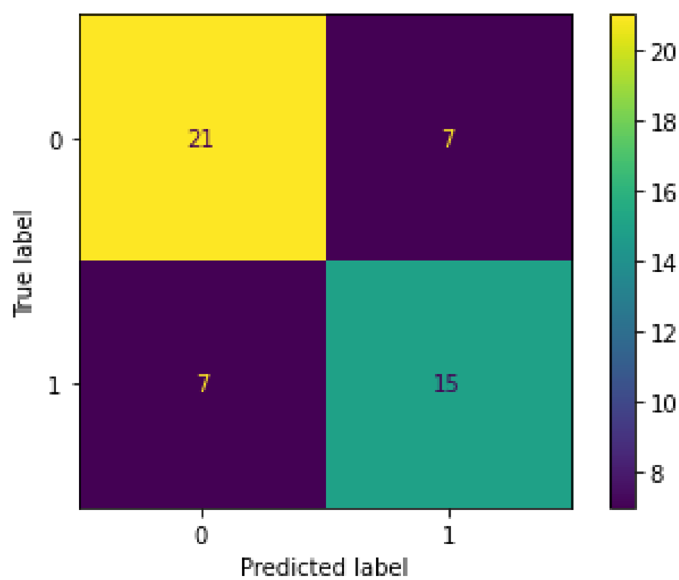
```
bayes_classifier = GaussianNB()
bayes_classifier.fit(X_train[50:], y_train[50:])
bayes_predictions = bayes_classifier.predict(X_train[:50])
plot_confusion_matrix(bayes_classifier, X_train[:50], y_train[:50])
acc_bayes = round(bayes_classifier.score(X_train[:50], y_train[:50]) * 100, 2)
```



ACCURACY : 78 %

LINEAR SVC

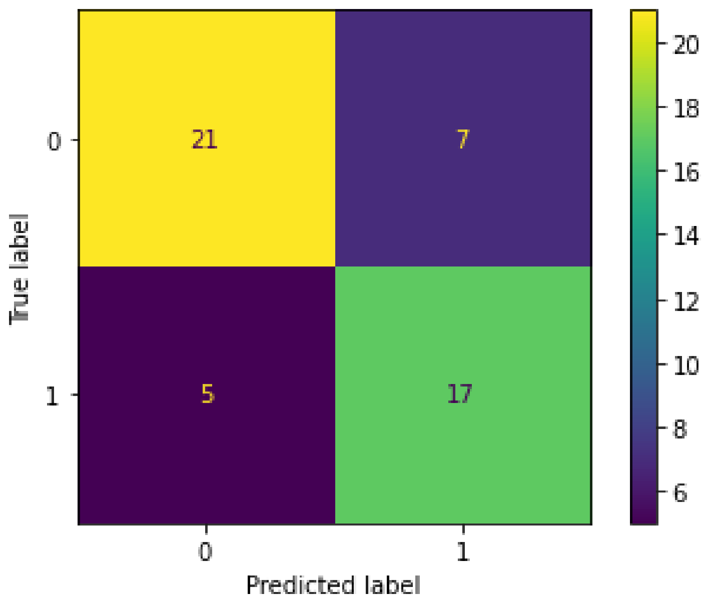
```
svc = LinearSVC()
svc.fit(X_train[50:], y_train[50:])
Y_pred_svm = svc.predict(X_test)
acc_linear_svc = round(svc.score(X_train[:50], y_train[:50]) * 100, 2)
```



ACCURACY: 72%

LOGISTIC REGRESSION

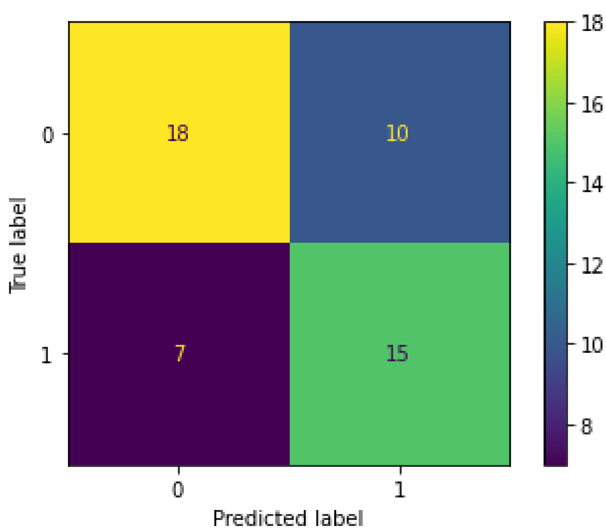
```
logreg = LogisticRegression()
logreg.fit(X_train[50:], y_train[50:])
Y_pred_log = logreg.predict(X_test)
acc_log = round(logreg.score(X_train[:50], y_train[:50]) * 100, 2)
```



ACCURACY: 76%

DECISION TREE

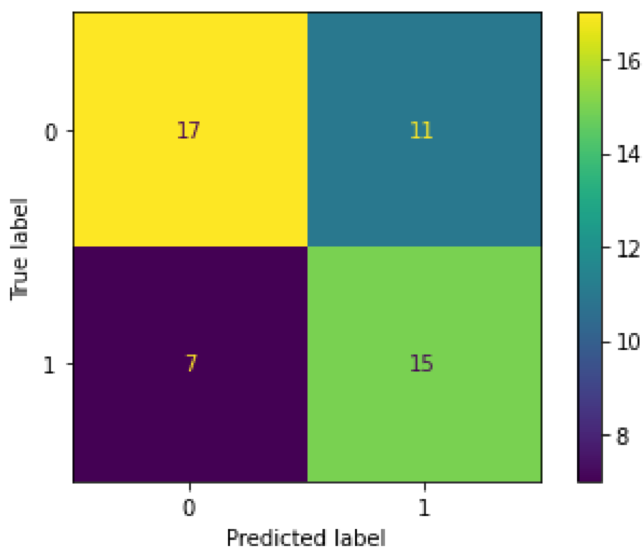
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train[50:], y_train[50:])
Y_pred_dt = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train[:50], y_train[:50]) * 100, 2)
```



ACCURACY: 66%

RANDOM FOREST

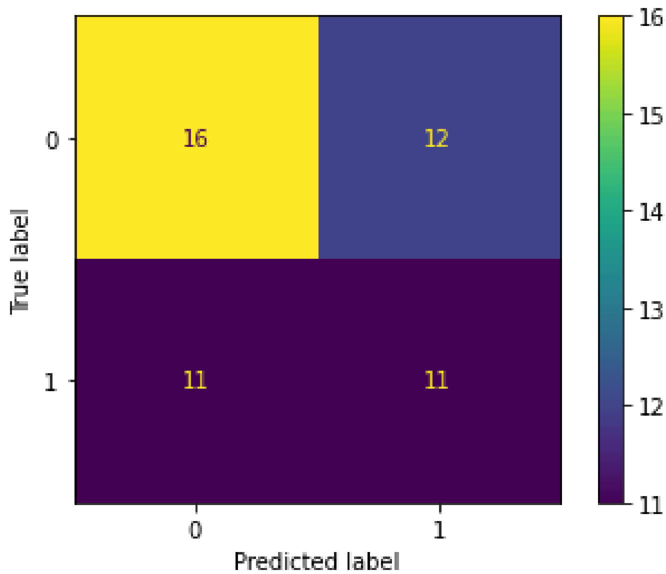
```
rf_classifier = RandomForestClassifier(n_estimators = 25)
rf_classifier.fit(X_train[50:], y_train[50:])
rf_predictions = rf_classifier.predict(X_test)
acc_random_forest = round(rf_classifier.score(X_train[:50], y_train[:50]) * 100, 2)
```



ACCURACY: 64%

K- NEAREST NEIGHBOUR

```
knn_classifier = KNeighborsClassifier(n_neighbors = 3)
knn_classifier.fit(X_train[50:], y_train[50:])
knn_predictions = knn_classifier.predict(X_train[:50])
plot_confusion_matrix(knn_classifier, X_train[:50], y_train[:50])
acc_knn = round(knn_classifier.score(X_train[:50], y_train[:50]) * 100, 2)
```



ACCURACY: 54%

NEURAL NETWORK

```
Model = Sequential()
Model.add(Dense(16,input_dim=(8),activation='relu'))
Model.add(Dense(1,activation='sigmoid'))
```

```
Model.summary()
```

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_42 (Dense)             (None, 16)                144
_________________________________________________________________
dense_43 (Dense)             (None, 1)                 17
=================================================================
Total params: 161
Trainable params: 161
Non-trainable params: 0
_________________________________________________________________
```

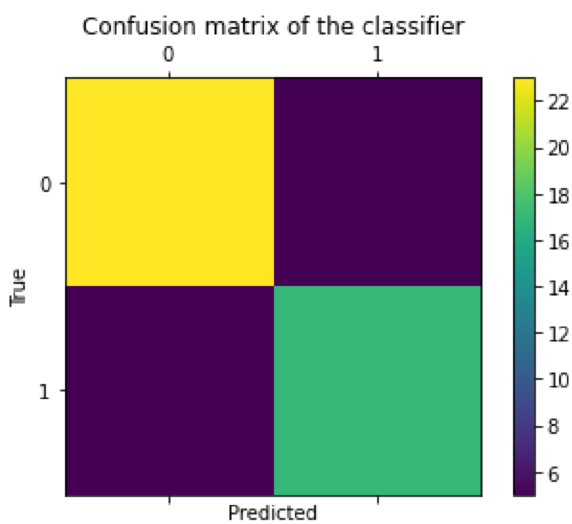
```
Model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
history=Model.fit(X_train[50:],y_train[50:],epochs=52,batch_size=32)
```

```
Model.evaluate(X_train[:50],y_train[:50])
```

```
2/2 [==============================] - 0s 2ms/step - loss: 0.4877 - accuracy:
0.8200

[0.48766040802001953, 0.8199999928474426]
```



ACCURACY : 82%

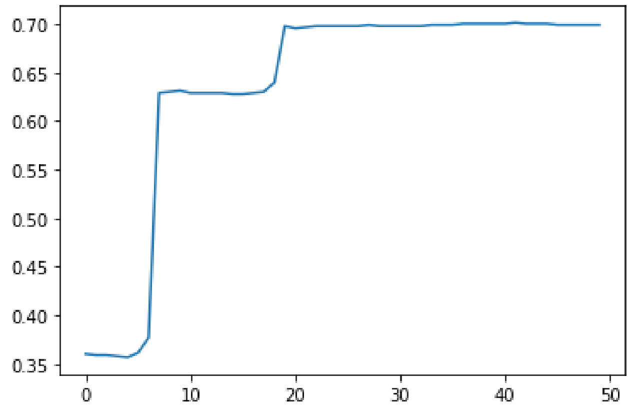


Figure 1: Progression of Accuracy of the Model

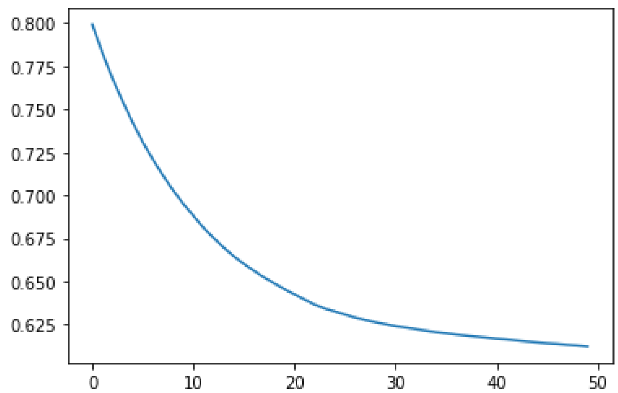


Figure 2:Progression of Loss of the Model

TABLE COMPARSION BETWEEN PREDICTION MODELS

Model	Score
Neural Network	82.000001
Naive Bayes	78.000000
Logistic Regression	76.000000
Linear SVC	72.000000
Decision Tree	66.000000
Random Forest	64.000000
KNN	54.000000

CONCLUSION

Based on our analysis and comparison of the prediction models it is evident that the Neural Network provides the best results for classification. This is due to the powerful generalization capability of Neural networks.

REFERENCES

[1] https://www.kaggle.com/c/titanic

[2] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

[3] https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/

[4] https://towardsdatascience.com/solving-a-simple-classification-problem-with-python-fruits-lovers-edition-d20ab6b071d2

[5] https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/