**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 4](#_Toc60765906)

[OVERVIEW OF THE DATA 4](#_Toc60765907)

[**FIRST FIVE ROWS OF THE TRAIN.CSV DATA** 4](#_Toc60765908)

[**FIRST FIVE ROWS OF THE TEST.CSV DATA** 4](#_Toc60765909)

[**FINDINGS BASED ON THE DATA** 5](#_Toc60765910)

[COMPARISON OF DIFFERENT PREDICTION MODELS 6](#_Toc60765911)

[**CODE** 6](#_Toc60765912)

[Importing the ModuLes 6](#_Toc60765913)

[Importing Data From CSV 7](#_Toc60765914)

[Removing Outliers from the Data 7](#_Toc60765915)

[SPlitting Independent and Dependent Variables 7](#_Toc60765916)

[Converting Test Data into Numpy Array 7](#_Toc60765917)

[Converting Gender From CATEGORICAL TO BINARY VARIABLE 8](#_Toc60765918)

[8](#_Toc60765919)

[FILLING MISSING VALUES OF EMBARKED WITH MODE 8](#_Toc60765920)

[FILLING MISSING AGE VALUES WITH MEAN AGE 8](#_Toc60765921)

[FILLING MISSING FARE VALUES WITH MEAN 8](#_Toc60765922)

[OneHot encoding passenger class 8](#_Toc60765923)

[SKIPPING DUMMY VARIABLE TRAP 9](#_Toc60765924)

[CONVERTING EMBARKED LOCATION TO SPARSE MATRIX 9](#_Toc60765925)

[APPLYING Z SCORE NORMALIZATION TO AGE 9](#_Toc60765926)

[APPLYING Z SCORE NORMALIZATION TO FARE 9](#_Toc60765927)

[APPLYING PCA FOR FEATURE EXTRACTION 10](#_Toc60765928)

[naïve BAYES CLASSIFER USING GAUSSIAN naïve BAYES 10](#_Toc60765929)

[LINEAR SVC 11](#_Toc60765930)

[LOGISTIC REGRESSION 11](#_Toc60765931)

[DECISION TREE 12](#_Toc60765932)

[RANDOM FOREST 13](#_Toc60765933)

[K- NEAREST NEIGHBOUR 14](#_Toc60765934)

[NEURAL NETWORK 14](#_Toc60765935)

[**TABLE COMPARSION BETWEEN PREDICTION MODELS** 16](#_Toc60765936)

[CONCLUSION 17](#_Toc60765937)

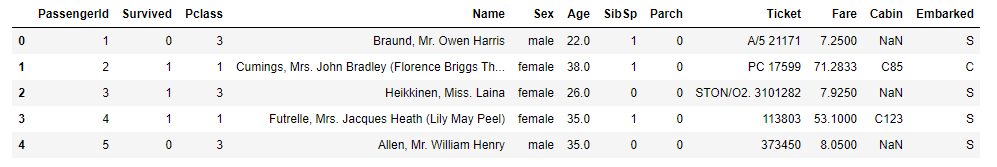
# Introduction

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.

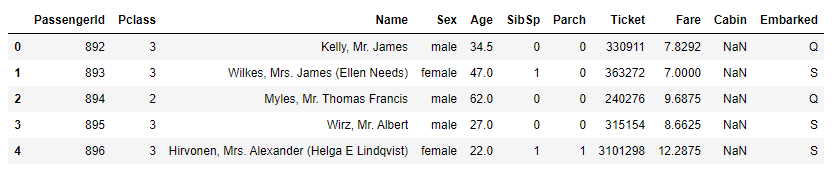
The train.csv file has 12 columns and 891 rows while the test.csv file has 11 columns and 418 rows.

# OVERVIEW OF THE DATA

## **FIRST FIVE ROWS OF THE TRAIN.CSV DATA**



## **FIRST FIVE ROWS OF THE TEST.CSV DATA**

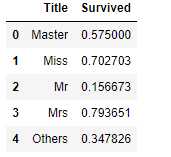


## **FINDINGS BASED ON THE DATA**

* 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:

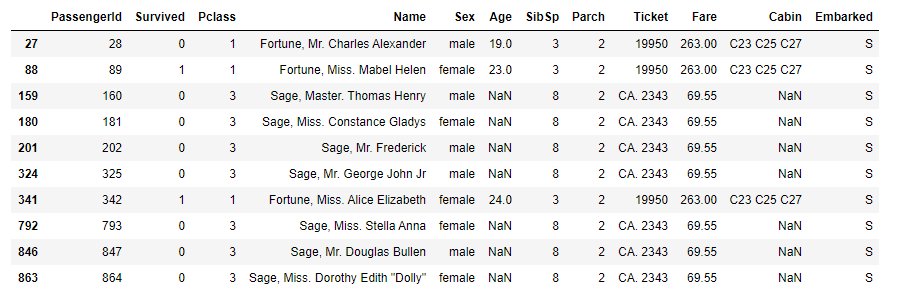
|  |  |
| --- | --- |
| **Cabin** | **687** |
| **Age** | **177** |
| **Embarked** | **2** |

* Survival rate based on the Title of the Passengers:

****

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:

****

# 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)

X\_test = ct\_pclass.transform(X\_test)

### SKIPPING DUMMY VARIABLE TRAP

ct\_pclass = ColumnTransformer([('one\_hot\_encoder', OneHotEncoder(categories='auto'), [0])],remainder='passthrough')

X\_train = ct\_pclass.fit\_transform(X\_train)

X\_test = ct\_pclass.transform(X\_test)

### 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

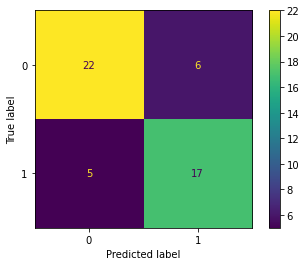
bayes\_classifier = GaussianNB()

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 %

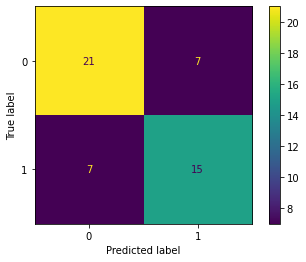
### LINEAR SVC

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%

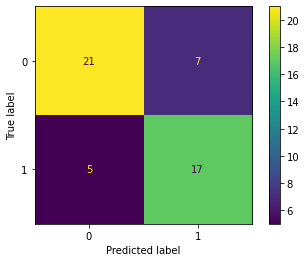
### LOGISTIC REGRESSION

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%

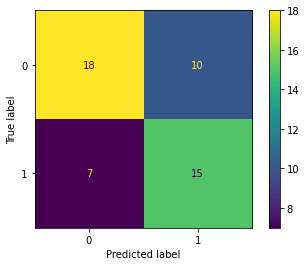
### DECISION TREE

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%

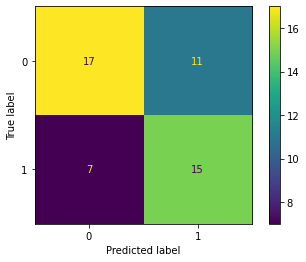
### RANDOM FOREST

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

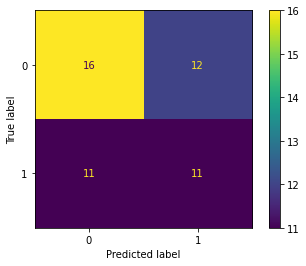
knn\_classifier = KNeighborsClassifier(n\_neighbors = 3)

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%

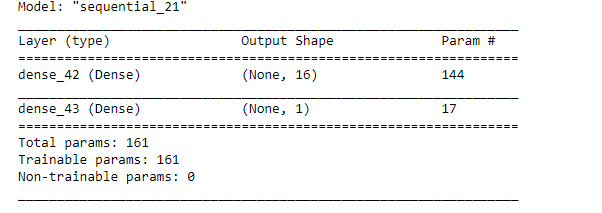
### NEURAL NETWORK

Model = Sequential()

Model.add(Dense(16,input\_dim=(8),activation='relu'))

Model.add(Dense(1,activation='sigmoid'))

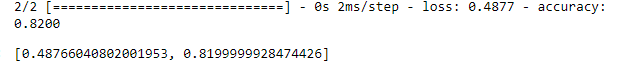
Model.summary()

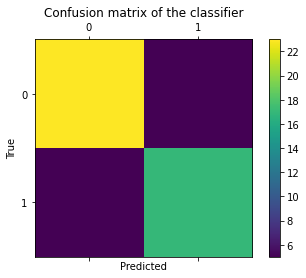


Model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

history=Model.fit(X\_train[50:],y\_train[50:],epochs=52,batch\_size=32)

Model.evaluate(X\_train[:50],y\_train[:50])





ACCURACY : 82%

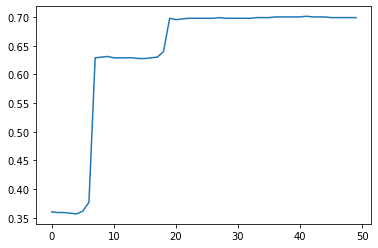


Figure 1: Progression of Accuracy of the Model

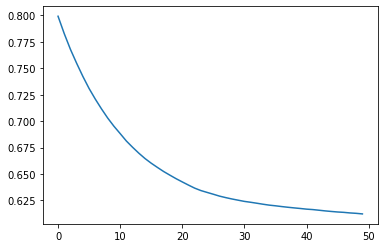
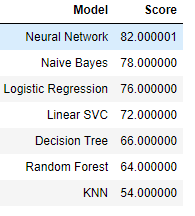


Figure 2:Progression of Loss of the Model

## **TABLE COMPARSION BETWEEN PREDICTION MODELS**



# 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/>