**ARTIFICIAL INTELLIGENCE**

**PROJECT**



**ANALYSIS OF TITANIC DATASET**

**USING DECISION TREE, K-NEAREST NEIGHBOUR AND RANDOM FOREST**

|  |  |
| --- | --- |
| **NAME** | **ROLL NO.** |
| PORUSH GOYAL | 17CSU136 |
| PARAS | 17CSU132 |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Name of the Topic** | **Page No.** |
|  | INTRODUCTION | 3 |
|  | DECISION TREE | 4 |
|  | K-NEAREST NEIGHBOR | 5 |
|  | RANDOM FOREST | 6 |
|  | IMPLEMENTATION | 7 |
|  | OUTPUT | 10 |
|  | BIBLIOGRAPHY | 14 |

**INTRODUCTION**

Titanic dataset has been chosen for the analysis. In this analysis the dataset has been analysed so as to predict the port of embarkation by generating a prediction model using all the relevant attributes in the dataset. In order to generate a model, data needed to be pre-processed so as to remove the columns that are not required, standardise all the features, filling the NAN values with the average value of the column. Mapping the categorical data to numeric value in order to improve the prediction accuracy of the model.

After the pre-processing is done the data was split into two groups training data and testing data. Training data is used for the creation of the model while the testing data is used to check the validity of the model.

Two algorithms namely Decision Tree, K-Nearest Neighbour and Random Forest were applied and the accuracies of the models generated from these algorithms was compared with each other in order to select the most suitable one for the analysis of the dataset.

**DECISION TREE**

* Decision tree induction is the learning of decision trees from class-labeled training tuples.
* A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or *terminal node*) holds a class label.
* The topmost node in a tree is the root node.

**Why Decision Tree Are So Popular**

* The construction of decision tree classifiers does not require any domain knowledge
* Appropriate for exploratory knowledge discovery.
* They can handle multidimensional data
* Representation of acquired knowledge in tree form is intuitive
* Learning & Classification steps of decision tree are simple & fast.
* Good Accuracy Model

**K-NEAREST NEIGHBOR**

* KNN is a non-parametric and lazy learning algorithm.
* Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real-world datasets do not follow mathematical theoretical assumptions.
* Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.
* KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry.
* KNN performs better with a lower number of features than a large number of features. Increase in dimension also leads to the problem of overfitting.
* KNN has the following basic steps:
  + Calculate distance
  + Find closest neighbors
  + Vote for labels

**RANDOM FOREST**

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

**Approach:**

* Pick at random K data points from the training set.
* Build the decision tree associated with those K data points.
* Choose the number N tree of trees you want to build and repeat step 1 & 2.
* For a new data point, make each one of your N tree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

**IMPLEMENTATION**

1. **Including all the libraries:**

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn import datasets

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.metrics import matthews\_corrcoef

from sklearn.metrics import recall\_score

from sklearn.model\_selection import train\_test\_split

from sklearn import tree

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

1. **Importing the csv file:**

df = pd.read\_csv('titanic.csv')

1. **Pre-Processing Data:**

df.isnull().sum()

df = df.drop("Cabin", axis = 1)

class\_le = LabelEncoder()

sex\_map = {"male":1, "female":0}

df['Sex'] = df['Sex'].map(sex\_map)

mean\_age = df["Age"].mean()

mean\_fare = df["Fare"].mean()

df['Age'].fillna(mean\_age, inplace = True)

df['Fare'].fillna(mean\_fare, inplace = True)

1. **Visualizing Data:**

for x in [1,2,3]: ## for 3 classes

df.Age[df.Pclass == x].plot(kind="kde")

plt.title("Age wrt Pclass")

plt.legend(("1st","2nd","3rd"))

plt.show()

sns.factorplot(x="Age", y="Embarked",

hue="Sex", row="Pclass",

data=df,

orient="h", size=2, aspect=3.5,kind = 'violin', split = True)

1. **Decision Tree:**

x = df[["Pclass","Sex","Age","SibSp","Parch","Fare"]]

y = df["Embarked"]

sns.heatmap(x.corr(), cmap='magma', linecolor='white', linewidth=1, annot = True)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y,test\_size = 0.2, random\_state=6)

Decision\_tree = tree.DecisionTreeClassifier(random\_state = 2)

Decision\_tree.fit(x\_train,y\_train)

predictions = Decision\_tree.predict(x\_test)

print(accuracy\_score(y\_test, predictions))

print('confusion matrix')

print(confusion\_matrix(y\_test, predictions))

print('Precision')

print(classification\_report(y\_test, predictions))

print('Recall')

print(recall\_score(y\_test, predictions,average = 'macro'))

1. **Genarating Pairplot:**

class\_le = LabelEncoder()

df["Embarked"] = class\_le.fit\_transform(df["Embarked"].values)

sns.pairplot(df,

x\_vars = ['Pclass','Sex','Age','SibSp','Parch'],

y\_vars = 'Embarked', kind="reg")

1. **K-Nearest Neighbor:**

x = df[["Pclass","Age","SibSp","Parch","Fare"]]

y = df["Embarked"]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y,test\_size = 0.2, random\_state =6)

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.transform(x\_test)

classifier = KNeighborsClassifier(n\_neighbors = 13)

classifier.fit(x\_train, y\_train) prediction\_2 = classifier.predict(x\_test)

print("Confusion Matrix")

print(confusion\_matrix(y\_test, prediction\_2))

print("Accuracy")

print(accuracy\_score(y\_test, prediction\_2))

1. **Random Forest:**

x = df[["Pclass","Age","SibSp","Parch","Fare"]]

y = df["Embarked"]

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,random\_state =6)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 17, random\_state = 1)

classifier.fit(x\_train, y\_train)

prediction\_3 = classifier.predict(x\_test)

print("Accuracy")

acc\_3 = accuracy\_score(y\_test, prediction\_3)

print(accuracy\_score(y\_test, prediction\_3),'\n')

print("Confusion Matrix")

print(confusion\_matrix(y\_test, prediction\_3),'\n')

print("Classification Report")

print(classification\_report(y\_test, prediction\_3))

1. **Comparing Algorithms:**

results = pd.DataFrame(columns = ['Decision Tree', 'K-NN','Random Forest'], index = ['Accuracy'])

results['Decision Tree'] = acc\_1

results['K-NN'] = acc\_2

results['Random Forest'] = acc\_3

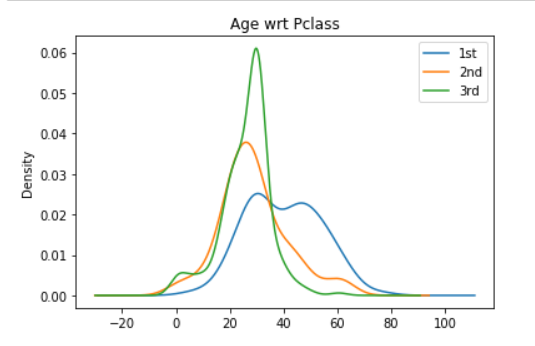
results

results.plot.bar()

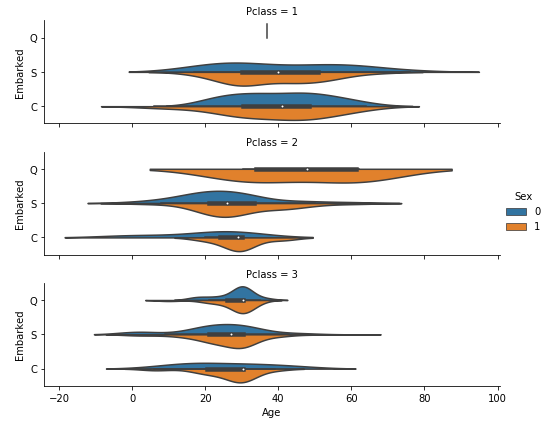
**OUTPUT**

1. **Data Visualization:**

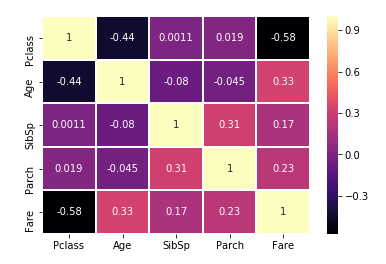
Depicting Age density with respect to class



Depicting Age distribution for each class form the respective port of embarkation

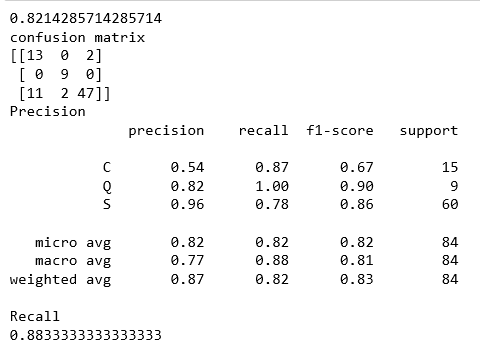


Heat Map shown correlation between variables used for generating the model

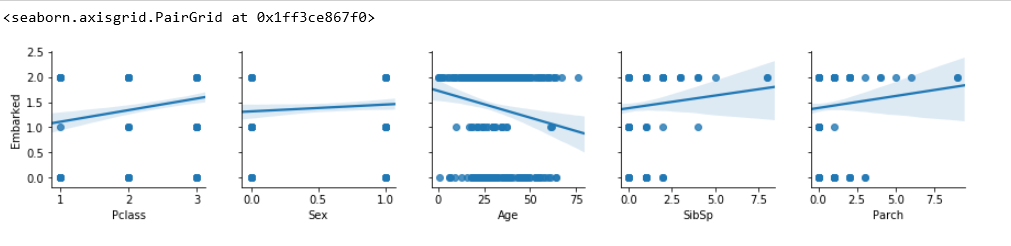


1. **Decision Tree:**

Results derived from the model

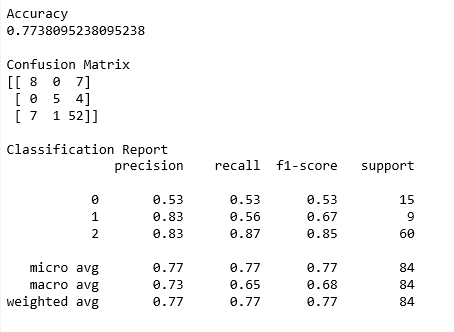


Pair Plot showing regression line between the independent variable and dependent variable “Embarked”



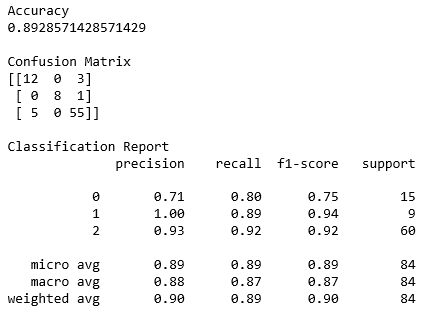
1. **K-Nearest Neighbor:**

Results generated from KNN model



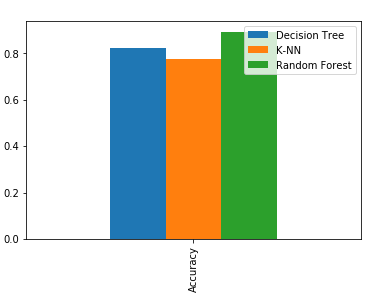
1. **Random Forest:**

Results obtained from random forest model



1. **Comparison of Algorithms:**

Bar plot used to depict accuracy of the 3 models



**BIBLIOGRAPHY**

* <https://www.kaggle.com/>
* <https://www.geeksforgeeks.org/>