**Decision Making In Event Management Using Data Science**

***Mini Project Report***

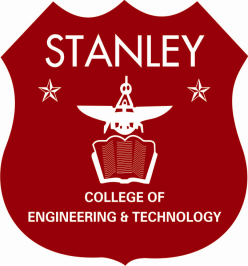
Submitted in partial fulfillment of the III year (Semester-I) BE degree as per the requirements of Osmania University,Hyderabad

By

Sravanthi Lakshmi Gunampalli 160620747046

Sidra Azeem 16062074744

Shamitha Jajula 160620747018



Department of ADCE

**Stanley College of Engineering and Technology for Women**

(Affiliated to OU & Approved by AICTE)

Chapel Road, Abids, Hyderabad - 500001

2022-2023



**Table of contents**

Abstract

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Study of existing system

1.4 Proposed System

2.System Requirements

3. Data Science Process

3.1 Data Mining

3.2 Data Cleaning

3.3 Data Exploration

3.4 Predictive Modeling

3.5 Data Visualization

4. Code Templates

5. Model validation

5.1 Prediction on Dataset

5.2 Accuracy

5.3 ROC & AUC

6. Output Screens

7. Conclusion

8. References/Bibliography

**ABSTRACT**

The study endeavor focuses on an analysis of successful events and event planning and marketing strategies. Specific methods will be looked at and determined in accordance with the events being analyzed, the organizational structure, and the anticipated results. Organizing events like concerts, shows, festivals, etc. Are pretty effective ways to raise money or create awareness for a cause. The efficiency strategies used by a company can help or hinder the sustainability of its goals and objectives. Organizations trying to generate money encounter a number of challenges as well as several techniques for getting money. Special events can help organizations find funding and further their core objectives. Special event offers a nonprofit organization significant community recognition by utilizing a captive audience of potential donors who are involved in the arts. Statistical methods and machine learning algorithms will be used to analyze the impact of parameters like date, time and place on the attendance for an event and draw meaningful information which will help estimate the success of a particular event, as well as improve decision-making when it comes to organizing an event. KNN (k-nearest neighbors), MLP (multi-layer perceptron), and Decision Tree classification models will be fitted for this dataset and their respective accuracies, ROC curves, and AUCs were compared to finalize a predictive model with customized parameters.

**INTRODUCTION**

“Humans are the most socially evolved species on the planet. As such, we develop social interactions beyond our family structures which can trigger the need for events” (Bladen et al. 2012, 7). Events create opportunities for people to connect with an area, spend time together, celebrate and experience the diversity of cultures and foster creativity and innovation. They allow a community to come alive and provide an opportunity for a destination to showcase its tourism experience and increase economic activity. Events contribute significantly to community building, lifestyle and leisure enhancement, cultural development, tourism promotion and increased visitation, volunteer participation, and economic development. Most importantly, events create a sense of fun and vibrancy, resulting in a strong sense of community connectivity, pride and a sense of place.

**1.1 Purpose**

This project will be a study on Successful Event Planning and Management. The primary aim of this study is to find out how managing and organising events is done in practice compared to what the literature says, and through the comparison to come across ideas for future developments of the event in question. The secondary aim of the study is for the authors to learn more about Event Management for future career possibilities. The intention is to study event management in aspects ranging from planning through implementation, to post-event evaluation. This is believed to provide the authors with relevant knowledge on how Event Planning and Management works in practice.The objectives of the Event Planning and Management System are to:

1. Enable those planning for events to do so easily, effectively and efficiently.
2. Give customers an attractive, logical shopping experience.
3. Give traders and suppliers a platform to advertise their goods and services online
4. Enable event planners to monitor what they have ordered online.
5. Save time and other valuable resources for those planning for events.

**1.2 Scope**

Understanding the power of data and how we can utilize it to derive results for the business by asking the right questions is more of an art than a science.

From the following pie charts, it can be understood that among all the event options present movie events need lesser expenditure and good average attendance, hence movies are a safer way to generate funds and raise awareness for a cause. Experience is an ally when organising an event, and for those with none...at the end of the project we might be able to offer that of a thousand.

Fig. 1.1

**1.3 Study of existing system**

Event planning is an uphill task for many people. As much as one would wish to have a perfect occasion, several factors are likely to hamper the process of event planning. One of these factors might be shrinky budgets. Anyone dreaming of a successful event should allocate adequate funds to have it sail through, but with the tough economic times, it’s become a challenge to utilize the limited resources. Time constraint is another major factor. Planning for an occasion needs a lot of time. Sometimes, time is consumed in consultations with different suppliers who offer various goods and services. Poor planning skills also will lead to a substandard event. This kind of poor planning is mainly brought about by the lack of exposure. When the planner is not well vast with event organizing, they tend to make numerous mistakes that may lead to a poorly planned event.

**1.4 Proposed System**

The objectives of the proposed system are:

* Analyze successful events and event planning and marketing strategies
* Draw meaningful information to improve decision-making when it comes to organizing an event
* Tackle afore mentioned problems using predictions and estimations
* Estimate the success of a particular event based on the interests of the target audience.

**SYSTEM REQUIREMENTS**

 An Intel-compatible platform running Windows 11, 10 /8.1/8 /7 /Vista /XP /2000 Windows Server 2022, 2019 /2016 /2012 /2008 /2003. At least 256 MB of RAM, a mouse, and enough disk space for recovered files, image files, etc.

**DATA SCIENCE PROCESS**

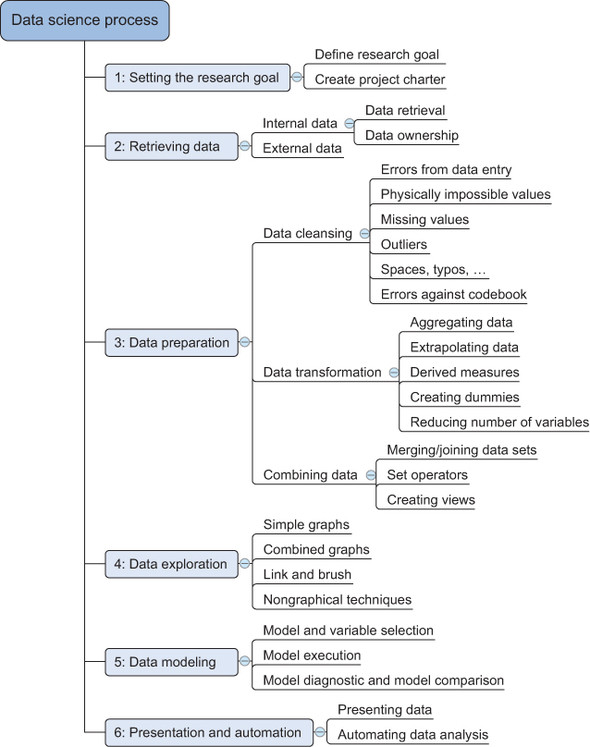


Fig.3.1

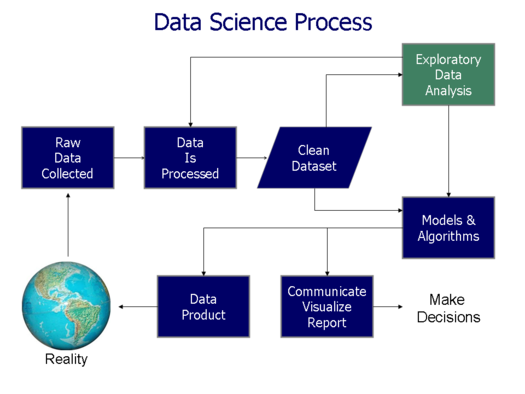


Fig. 3.2

**3.1 Data Mining**

Data mining is the process of gathering the data from different sources. We found a dataset with Event and attendance information from special events facilitated by NYC Parks' "Public Programs" division, including fitness, sports, dancing, movies, and concerts. This dataset had information about the number of people attending the events which were organized in the parks and recreation centers of New York on specific days of the weeks at different times of the day, on different dates in specific boroughs, location types and locations and the groups that conduct these events and the kind of public that attended the event.

The dataset had 12 attributes and 650 variables in total.

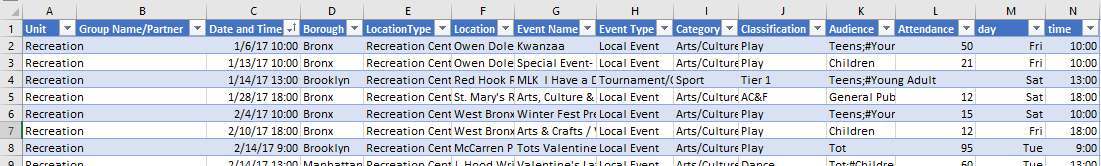


Fig. 3.1.1

Of these attributes, there was 1 numerical attribute, and the remaining were string/categorical attributes. From this dataset we subset the data related to the event Movies Under The Stars, for which we got 399 rows of data. MUTS is a family-friendly entertainment event in the city's parks and playgrounds, ranging from great new movies to all-time classics. Through Movies Under the Stars, the Mayor’s Office of Media and Entertainment and NYC Parks bring more than 150 film screenings to parks throughout the five boroughs.

**3.2 Data Cleaning**

Several excel manipulations and column selections later, we had the following dataset:

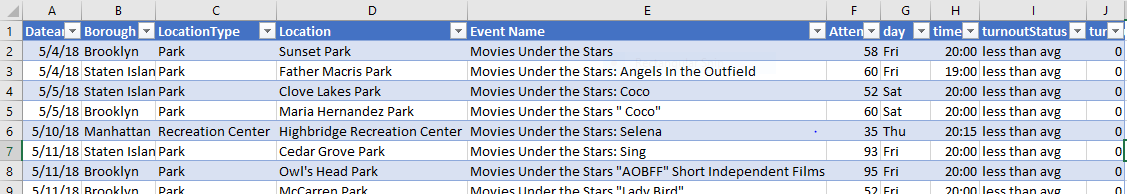


Fig. 3.2.1

We deleted the unit, public, group name, event type, category, audience columns and We extracted information about the time of the day and day of the week from the date time column and added 2 new columns to categorize the attendance into less than or more than average attendance, which was around 74 after outlier imputations.

Now that we’ve got all of the data, we move on to cleaning and preparing the data. This process can often take 50 to 80 percent of their time. Missing data can throw a lot of errors in the model creation and training. One option is to either ignore the instances which have any missing values. Depending on the dataset, this could be unrealistic if we have a lot of missing data. Another common approach is to use something called average imputation, which replaces missing values with the average of all the other instances. This is not always recommended because it can reduce the variability of the data. Luckily there was no missing data (to the extent that we used) in this dataset.

**3.3 Data Exploration**

The data exploration stage is like the brainstorming of data analysis. This is where we understand the patterns and bias in the data. It involves pulling up and analyzing a subset of the data using pandas, plotting a histogram or distribution curve to see the general trend, or even creating an interactive visualization that lets us dive down into each data point and explore the story behind the outliers.Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

Using all of this information, we started to form hypotheses about the data and the problem we are tackling.

We employed boxplot outlier zero value imputations to deal with the outliers in the dataset.

**3.4 Predictive Modeling**

Predictive modeling is where the machine learning finally comes into the data science project. Based on the questions we asked in the business understanding stage, this is where we decide which model to pick for the problem. This is never an easy decision, and there is no single right answer.

Once we’ve trained the model, it is critical that we evaluate its success. For classification models, we often test accuracy using pcc (percent correct classification), along with a confusion matrix which breaks down the errors into false positives and false negatives. Plots such as as roc curves, which is the true positive rate plotted against the false positive rate, are also used to benchmark the success of a model.

KNN (k-nearest neighbors), MLP (multi-layer perceptron), and Decision Tree classification models were fitted for this dataset and their respective accuracies, ROC curves, and AUCs were compared to finalize a predictive model.

**3.5 Data Visualization**

Data visualization is a tricky field, mostly because it seems simple but it could possibly be one of the hardest things to do well. That’s because data viz combines the fields of communication, psychology, statistics, and art, with an ultimate goal of communicating the data in a simple yet effective and visually pleasing way. Once we’ve derived the intended insights from the model, we represent them in way that the different key stakeholders in the project can understand.

**CODE TEMPLATES**

**LOADING PACKAGES**

import numpy as np

import pandas as pd

import sklearn

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import preprocessing

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics

from sklearn.metrics import accuracy\_score

from sklearn import datasets

from sklearn.metrics import roc\_curve, roc\_auc\_score

First, we imported several necessary packages in Python:

* numpy: NumPy is used for advanced array operations (e.g. Add, multiply, slice, reshape, index),comprehensive mathematical functions, random number generation, linear algebra routines, Fourier transforms, etc.
* pandas: pandas can be used for reading/writing data from/to csv and excel files and SQL databases, reshaping and pivoting datasets, slicing, indexing, and subsetting datasets, aggregating and transforming data, merging and joining datasets.
* sklearn: scikit-learn is an efficient and beginner-friendly tool for predictive data analysis. Among other things, you can use scikit-learn to identify which category an object is likely to belong to (used in fraud detection, image recognition, cancer detection, etc.), predict a continuous variable based on available features (used in predicting house prices and inflation), group similar objects into clusters (used in customer segmentation, social network analysis, etc.).
* seaborn: Seaborn is a high-level interface for drawing attractive statistical graphics with just a few lines of code.
* matplotlib: Matplotlib is the most common data exploration and visualization library. One can use it to create basic graphs like line plots, histograms, scatter plots, bar charts, and pie charts. One can also create animated and interactive visualizations with this library. Matplotlib is the foundation of every other visualization library.

**LOADING DATA**

data = pd.read\_csv("C:/Users/Sravanthi/Desktop/miniproject/parksrecmovies.csv")

data.head()

datran=pd.read\_csv("C:/Users/Sravanthi/Desktop/miniproject/prmtransform.csv")

Load the datasets that will be used in this program. It will be a CSV file with raw data. This dataset will be converted to pandas data frame once it is loaded.

**EDA**

data.dtypes

data.shape

data.describe()

data.info()

data.isnull().sum()

# histogram

sns.histplot(x='Attendance',y='day', data=data, )

plt.show()

sns.histplot(x='Attendance',y='time', data=data, )

plt.show()

# boxplot

sns.boxplot( x="Attendance", y='Borough', data=data, )

plt.show()

# scatter bivariate

sns.scatterplot( x="Attendance", y='Borough', data=data,

hue='turnoutStatus', size='Attendance')

# Placing Legend outside the Figure

plt.legend(bbox\_to\_anchor=(1, 1), loc=2)

plt.show()

We got a quick summary of the dataset using the describe() method. The describe() function applies basic statistical computations on the dataset like extreme values, count of data points standard deviation, etc. Any missing value or NaN value is automatically skipped. Describe() function gives a good picture of the distribution of data.

After that we got an idea about the dataset used and saw if our dataset contained any missing value or not. There weren’t any.

We used matplotlib and seaborn library for the data visualization and got the following plots on our dataset:

•histogram

A histogram was plotted between attendance and day attributes.

•boxplot

Boxplots were plotted between attendance and time, borough attributes.

•scatter plot

A scatter plot was plotted between attendance and borough attributes.

**DATA CLEANING**

#boxplot outlier imputations

sns.boxplot(data['Attendance'])

plt.title("Box Plot before imputation")

plt.show()

train = data.copy()

q1 = train['Attendance'].quantile(0.25)

q3 = train['Attendance'].quantile(0.75)

iqr = q3-q1

Lower\_tail = q1 - 1.5 \* iqr

Upper\_tail = q3 + 1.5 \* iqr

m = np.mean(train['Attendance'])

for i in train['Attendance']:

if i > Upper\_tail or i < Lower\_tail:

train['Attendance'] = train['Attendance'].replace(i, m)

sns.boxplot(train['Attendance'])

plt.title("Box Plot after mean imputation")

plt.show()

train = data.copy()

q1 = train['Attendance'].quantile(0.25)

q3 = train['Attendance'].quantile(0.75)

iqr = q3-q1

Lower\_tail = q1 - 1.5 \* iqr

Upper\_tail = q3 + 1.5 \* iqr

med = np.median(train['Attendance'])

for i in train['Attendance']:

if i > Upper\_tail or i < Lower\_tail:

train['Attendance'] = train['Attendance'].replace(i, med)

sns.boxplot(train['Attendance'])

plt.title("Box Plot after median imputation")

plt.show()

train = data.copy()

q1 = train['Attendance'].quantile(0.25)

q3 = train['Attendance'].quantile(0.75)

iqr = q3-q1

Lower\_tail = q1 - 1.5 \* iqr

Upper\_tail = q3 + 1.5 \* iqr

for i in train['Attendance']:

if i > Upper\_tail or i < Lower\_tail:

train['Attendance'] = train['Attendance'].replace(i, 0)

sns.boxplot(train['Attendance'])

plt.title("Box Plot after Zero value imputation")

plt.show()

print(train['Attendance'])

data['Attendance'] = data['Attendance'].replace(train['Attendance'])

# writing into the file

data.to\_csv("C:/Users/Sravanthi/Desktop/miniproject/parksrecmovies.csv", index=False)

print(data)

The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process in the data frame same as removing a data item from the panda’s data frame. After detection outliers in the boxplot, boxplot zero value imputations were applied on the attendance column and modifications were made to the dataset. 10% of the dataset was modified and the average attendance of MUTS event came down from 126 to 74.

**LABEL ENCODING**

# Creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers.

data['Borough']=le.fit\_transform(data['Borough'])

data['LocationType']=le.fit\_transform(data['LocationType'])

data['day']=le.fit\_transform(data['day'])

#data['turnoutStatus']=le.fit\_transform(data['turnoutStatus'])

data['time']=le.fit\_transform(data['time'])

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning. LabelEncoder() function was applied on the borough, location type, day, time attributes and the categorical values were converted into numerical values.

**SPLITTING DATA INTO FEATURE AND CLASS NAMES**

X=data[['Borough','LocationType','day','time']]

y=data['turnoutStatus123']

We took Borough, locationType, day, and time attribute variables as independent/predictor variables and turnoutStatus123 attribute variables as dependent/response variable.

**SPLITTING DATA INTO TRAINING AND TEST SET**

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42

The train\_test\_split() method is used to split the data into train and test sets. First, we need to divide our data into features (X) and labels (y). The dataframe gets divided into X\_train,X\_test , y\_train and y\_test. X\_train and y\_train sets are used for training and fitting the model. Here the dataset was split into 80% training and 20% test.

**MODEL FITTING**

MLP CLASSIFIER

# Create model object

clf = MLPClassifier(hidden\_layer\_sizes=(6,5),

random\_state=5,

verbose=True,

learning\_rate\_init=0.01)

clf.fit(X\_train,y\_train)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

KNN MODEL

knn = KNeighborsClassifier(n\_neighbors=7)

knn.fit(X\_train, y\_train)

DECISION TREE CLASSIFIER

tstree=DecisionTreeClassifier(criterion='entropy',max\_depth=4)

tstree

tstree.fit(x\_trainset,y\_trainset)

KNN(k-nearest neighbors), MLP(multi-layer perceptron) , and Decision Tree classification models were fitted for this dataset.

* KNN classifier

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

* MLP classifier

A multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. MLP uses back propagation for training the network.

* Decision tree classifier

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

**DATA VISUALIZATION**

DECISION TREE CLASSIFIER

fn=data.columns[1:5]

cn=data["turnoutStatus"].unique().tolist()

tstree.fit(X,y)

fig,axes=plt.subplots(nrows=1,ncols=1,figsize=(10,10),dpi=300)

tree.plot\_tree(tstree,feature\_names=fn,class\_names=cn,filled=True)

fig.savefig('prmacc.png')

print(tree.plot\_tree(tstree,feature\_names=fn,class\_names=cn,filled=True))

A decision tree was obtained for this dataset after the model was fit.

**DATA PRODUCT**

KNN

# Create feature and target arrays

X=data[['Borough','LocationType','day','time']]

y=data['turnoutStatus']

# Split into training and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size = 0.2, random\_state=42)

knn = KNeighborsClassifier(n\_neighbors=7)

knn.fit(X\_train, y\_train)

# Predict on dataset which model has not seen before

print(knn.predict(X\_test))

#predict

print(datran)

print("enter event parameters")

b=int(input("borough:"))

lt=int(input("location type:"))

d=int(input("day:"))

t=int(input("time:"))

x\_new=[[b,lt,d,t]]

pred=knn.predict(x\_new)

print("predicted attendance=",pred)

After comparing the 3 models, accuracy of KNN(0.835) & MLP(0.835) models was better than the accuracy of decision tree classifier(0.75) and the AUC of KNN(0.725) was higher than MLP(0.662), hence the KNN model was finalized as the predictive model for this project.

Now parameter related values for borough, location type, day, and time of the event to be organized can be entered to predict if the attendance to this event will be less than or more than average(74) using the KNN model.

**MODEL VALIDATION**

**5.1 PREDICTION ON DATASET**

MLP CLASSIFIER

# Make prediction on test dataset

ypred=clf.predict(X\_test)

print(ypred)

comparision = []

for i,j in zip(y\_test,ypred):

if i==j:

comparision.append(True)

else:

comparision.append(False)

print(comparision)

if all(comparision):

print('Both arrays are equal')

else:

print('Both Arrays are not equal')

KNN MODEL

# Predict on dataset which model has not seen before

print(knn.predict(X\_test))

DECISION TREE CLASSIFIER

predtree=tstree.predict(x\_testset)

print(predtree[0:5])

print(y\_testset[0:5])

print(predtree)

**5.2 ACCURACY**

MLP CLASSIFIER

# Calculate accuracy

print("accuracy=",accuracy\_score(y\_test,ypred))

KNN MODEL

# Predict on dataset which model has not seen before

print(knn.predict(X\_test))

print("accuracy:",metrics.accuracy\_score(y\_test,knn.predict(X\_test)))

DECISION TREE CLASSIFIER

print("accuracy:",metrics.accuracy\_score(y\_testset,predtree))

**5.3 ROC & AUC**

MLP CLASSIFIER

clf\_mlp = MLPClassifier(hidden\_layer\_sizes=(6,5),

random\_state=5,

verbose=True,

learning\_rate\_init=0.01);

clf\_mlp.fit(X\_train, y\_train);

y\_score1 = clf\_mlp.predict\_proba(X\_test)[:,1]

false\_positive\_rate3, true\_positive\_rate3, threshold1 = roc\_curve(y\_test, y\_score1)

print('roc\_auc\_score for MLP classifier: ', roc\_auc\_score(y\_test, y\_score1))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - MLP classifier')

plt.plot(false\_positive\_rate3, true\_positive\_rate3)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

KNN MODEL

#model validation

clf\_knn = KNeighborsClassifier(n\_neighbors=7);

clf\_knn.fit(X\_train, y\_train);

y\_score1 = clf\_knn.predict\_proba(X\_test)[:,1]

false\_positive\_rate2, true\_positive\_rate2, threshold1 = roc\_curve(y\_test, y\_score1)

print('roc\_auc\_score for KNN model: ', roc\_auc\_score(y\_test, y\_score1))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - KNN model')

plt.plot(false\_positive\_rate2, true\_positive\_rate2)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

DECISION TREE CLASSIFIER

#model validation

clf\_tree = DecisionTreeClassifier();

clf\_tree.fit(x\_trainset, y\_trainset);

y\_score1 = clf\_tree.predict\_proba(x\_testset)[:,1]

false\_positive\_rate1, true\_positive\_rate1, threshold1 = roc\_curve(y\_testset, y\_score1)

print('roc\_auc\_score for DecisionTree: ', roc\_auc\_score(y\_testset, y\_score1))

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic - DecisionTree')

plt.plot(false\_positive\_rate1, true\_positive\_rate1)

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

**KNN MODEL**

[0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0]

accuracy: 0.8375

roc\_auc\_score for KNN model: 0.7256027554535017

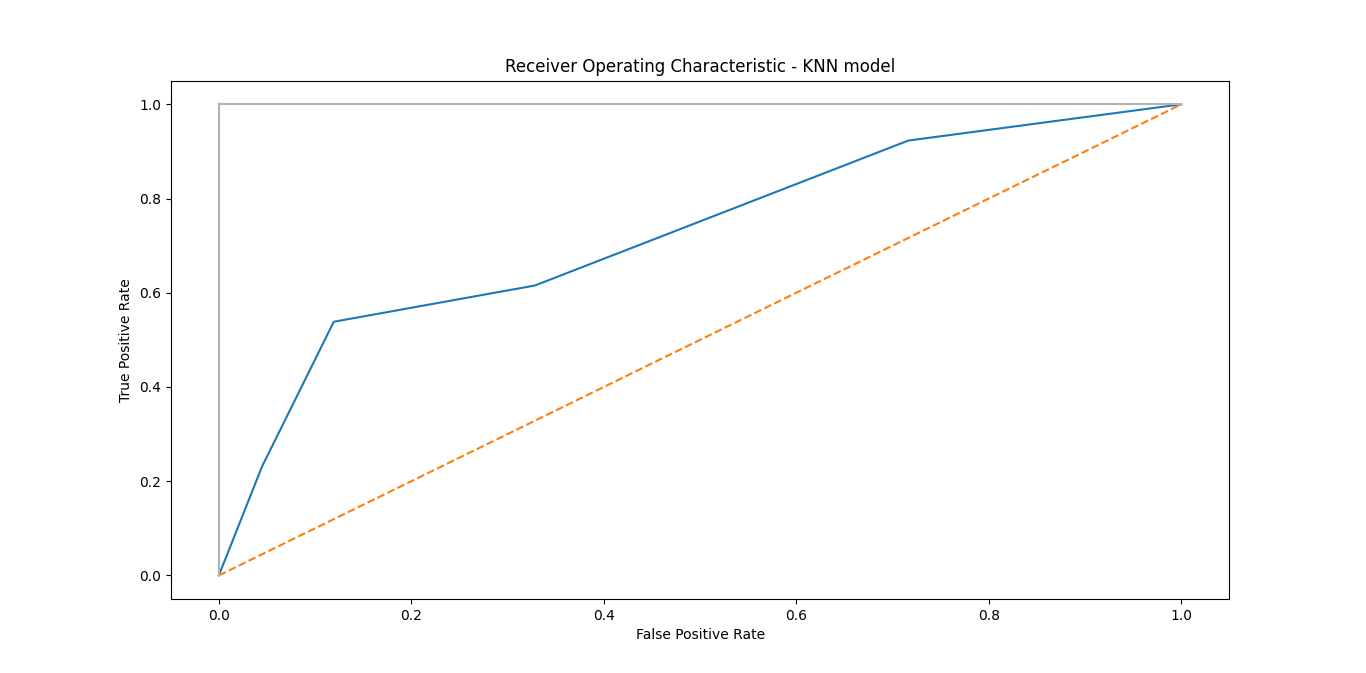


Fig. 5.1

**DECISION TREE CLASSIFIER**

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1]

accuracy: 0.75

roc\_auc\_score for DecisionTree: 0.5953408110440034

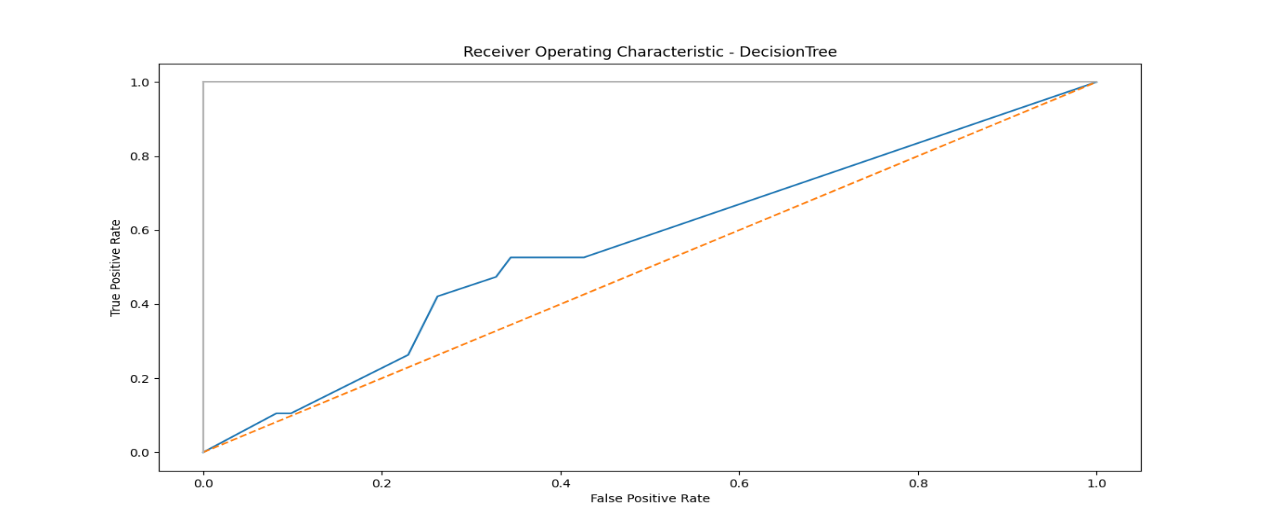


Fig. 5.2

**MLP CLASSIFIER**

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0]

[True, False, True, False, True, True, True, False, True, False, True, False, True, True, True, False, True, True, True, True, True, True, True, True, True, True, False, False, True, True, True, False, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True, True, True, True, True, False, True, False, True, True, True, True, True, True, True, True, True, True, True, False, True, True, True, True, True, True, True, True, True, True]

Both Arrays are not equal

accuracy= 0.8375

roc\_auc\_score for MLP classifier: 0.6624569460390356

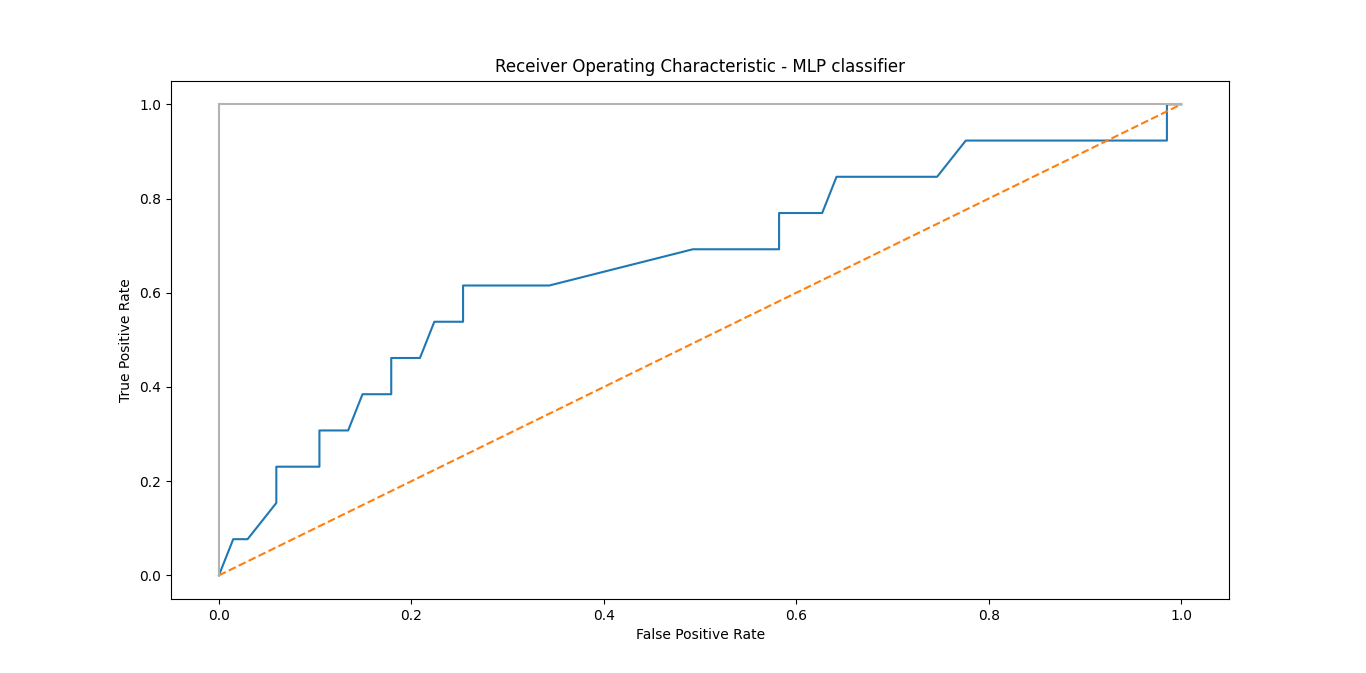


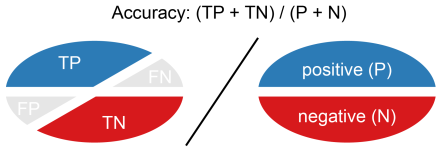
Fig. 5.3

The performance measures used to finalise a model were:

1. Accuracy
2. ROC curve
3. AUC

#### **Accuracy**

Accuracy (ACC) is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0.



Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

\mathrm{ACC = \displaystyle \frac{TP +TN}{TP + TN + FN + FP} = \frac{TP + TN}{P + N}}

* **ROC curve**

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

* True Positive Rate
* False Positive Rate

**True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

TPR=TPTP+FN

**False Positive Rate** (**FPR**) is defined as follows:

FPR=FPFP+TN

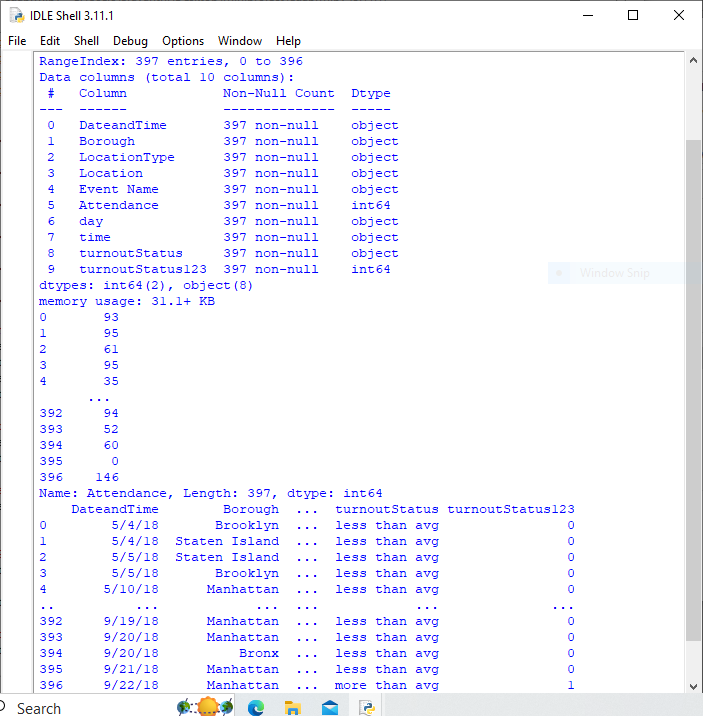
An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

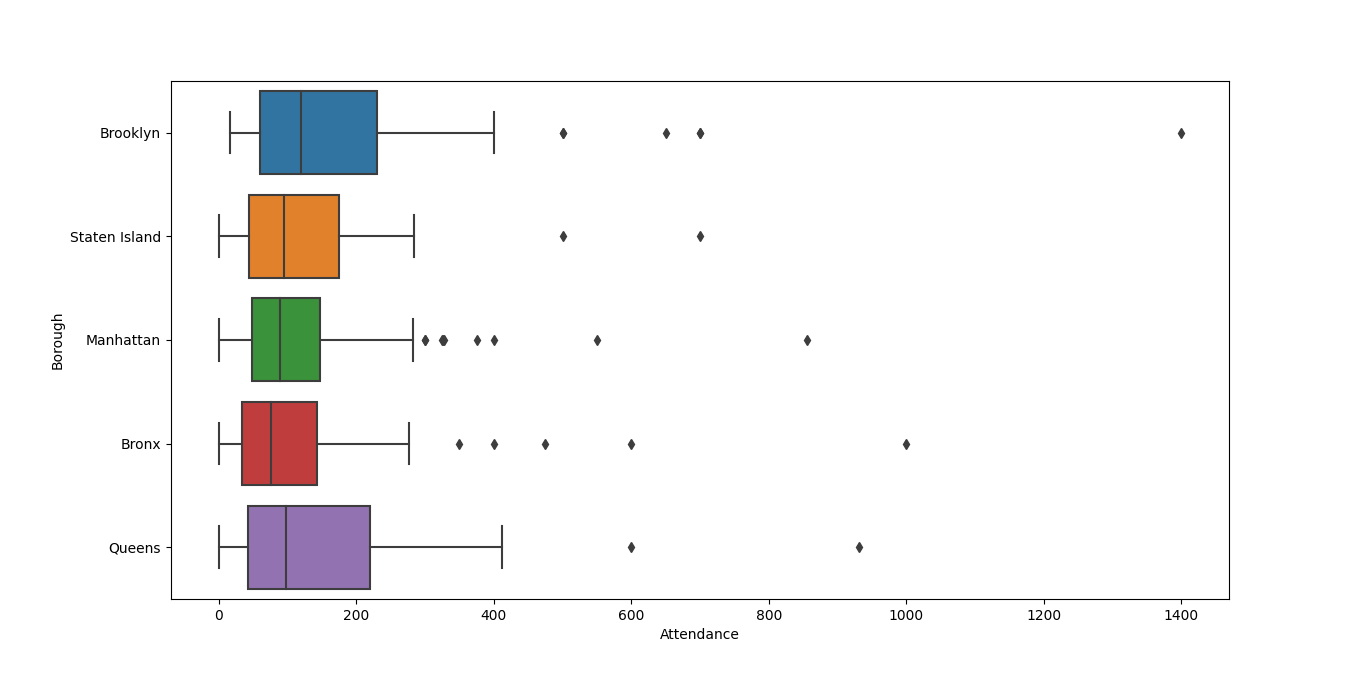
## **AUC: Area Under the ROC Curve**

**AUC** stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

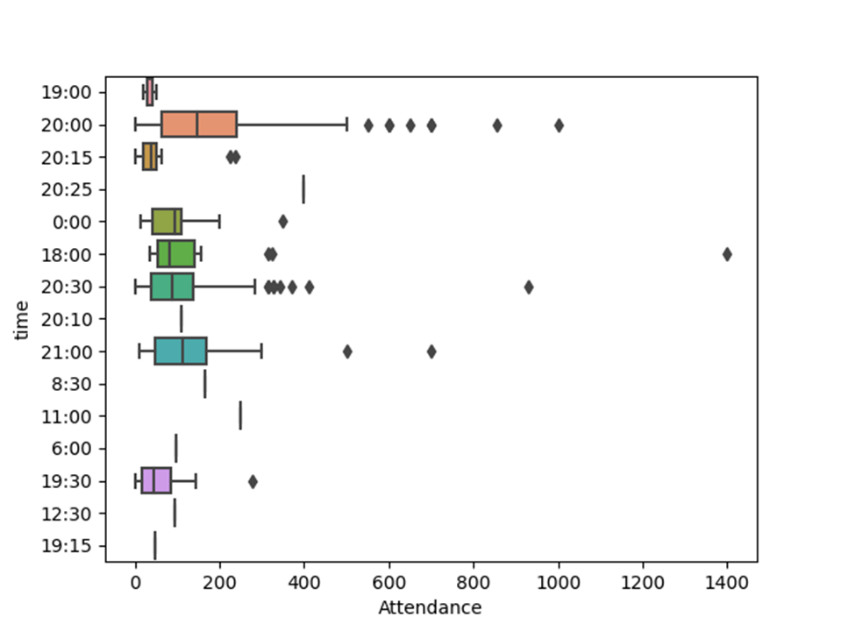
**OUTPUT SCREENS**

**6.1 EDA**

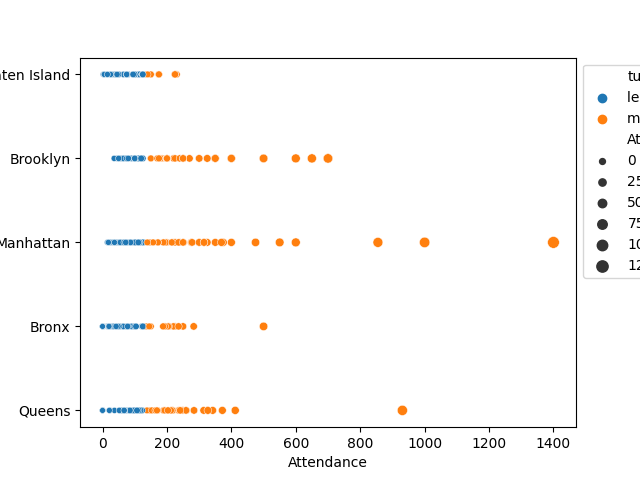


**6.1.1 BOXPLOT** 

**Fig. 6.1.1.1 Boxplot of Attendance vs Borough**

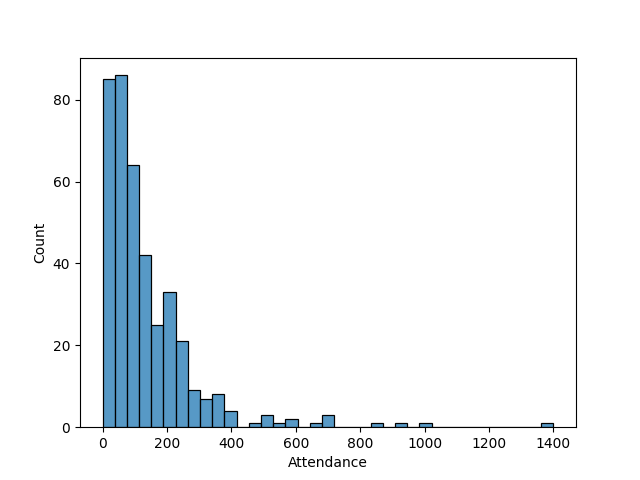


**Fig. 6.1.1.2 Boxplot of Attendance vs time**

**6.1.2 SCATTER PLOT**

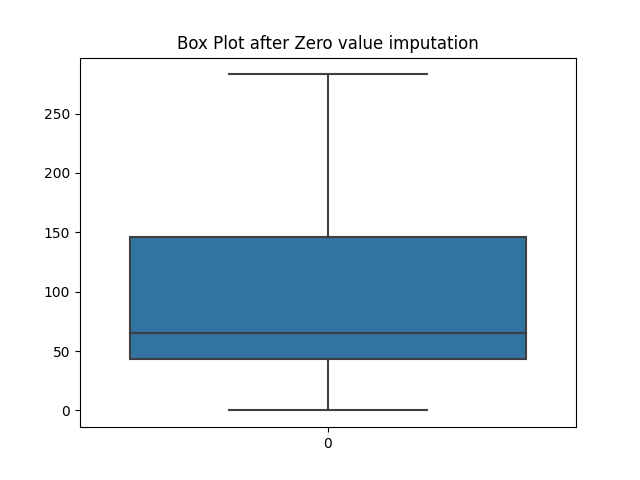
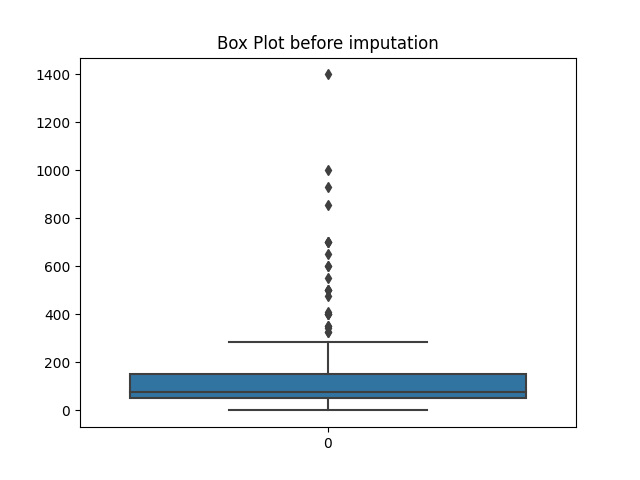
**Fig. 6.1.2 Scatter Plot of Attendance**

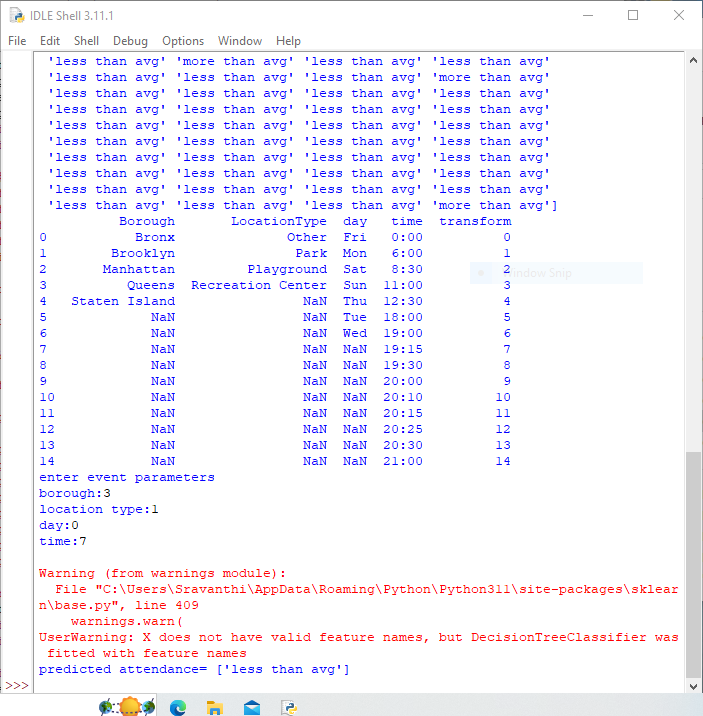
**6.1.3 Histogram**

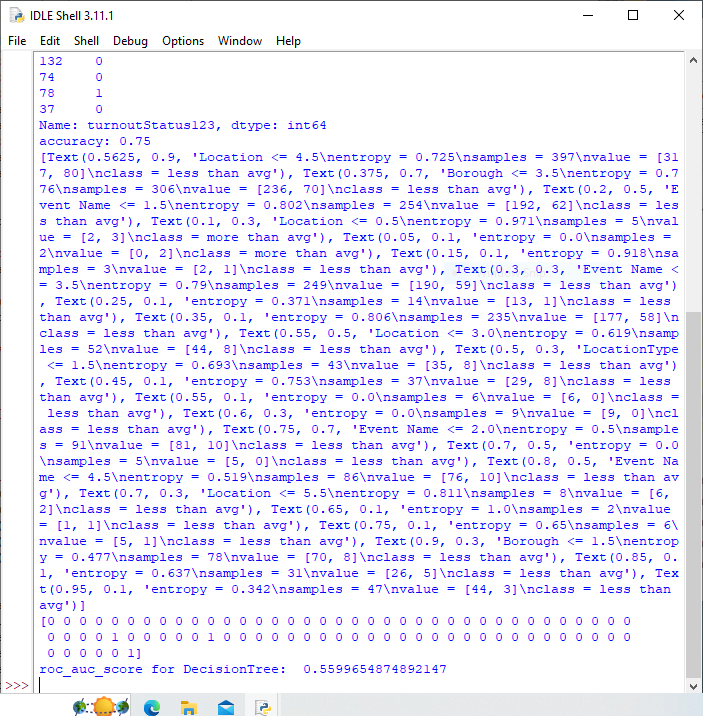


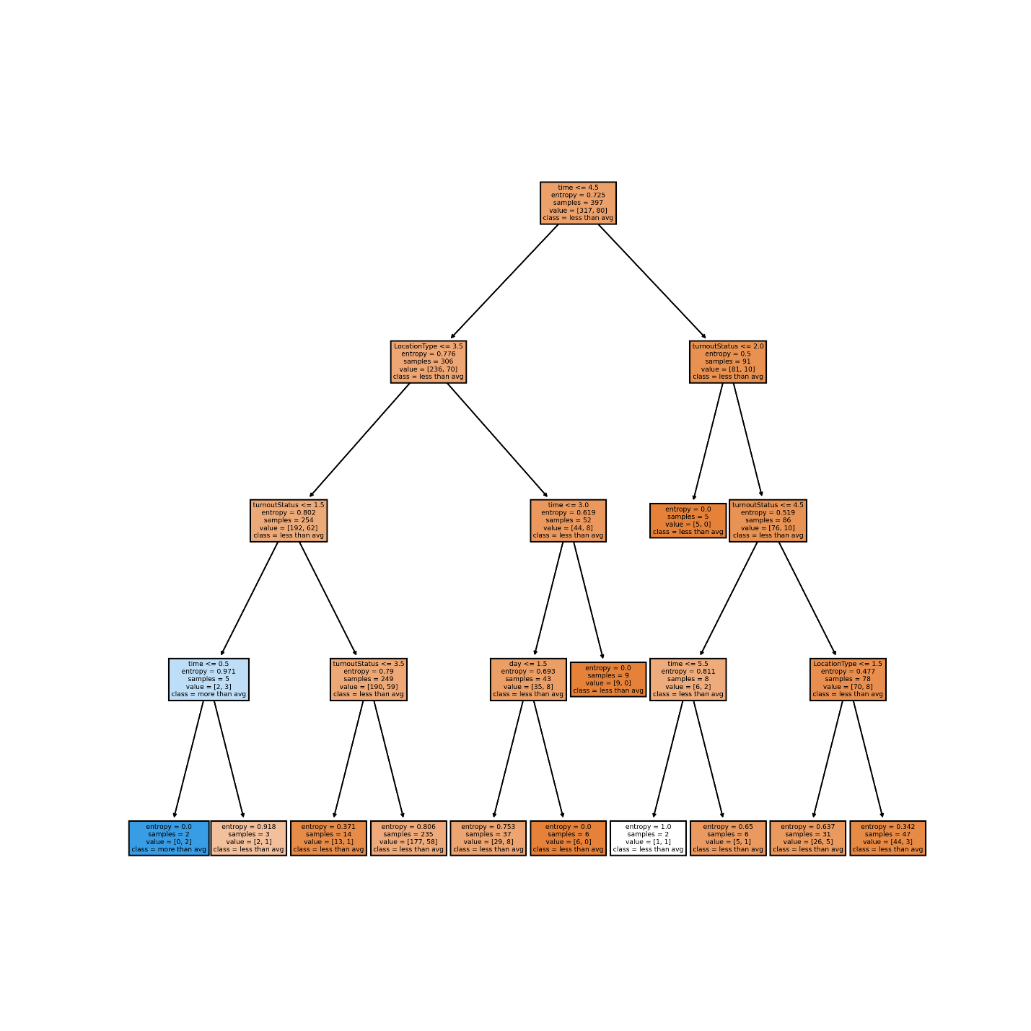
**Fig. 6.1.3 Histogram of Attendance vs count**

**6.2 OUTLIER HANDLING**



**6.3 KNN MODEL**

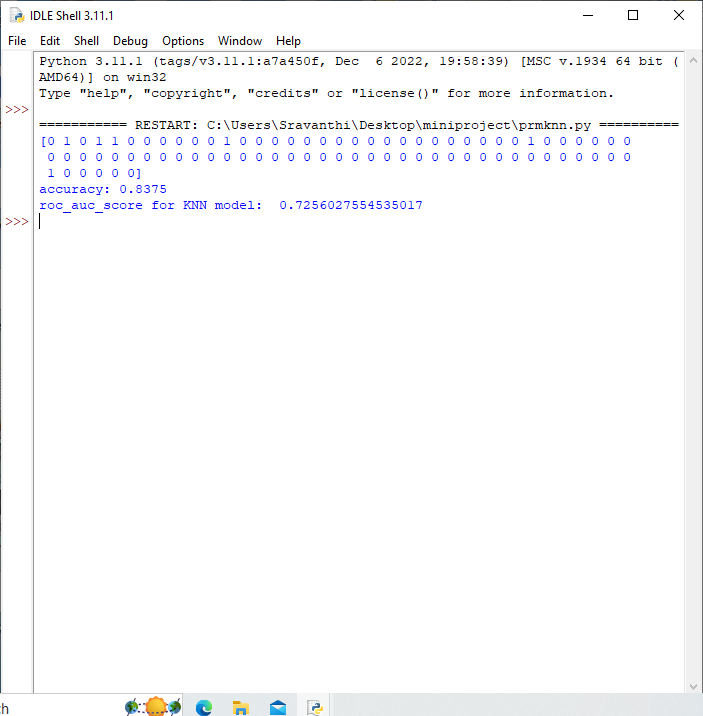
**6.4 DECISION TREE CLASSIFIER**



**Fig. 6.4.1 Decision Tree**

**6.5 MLP CLASSIFIER**

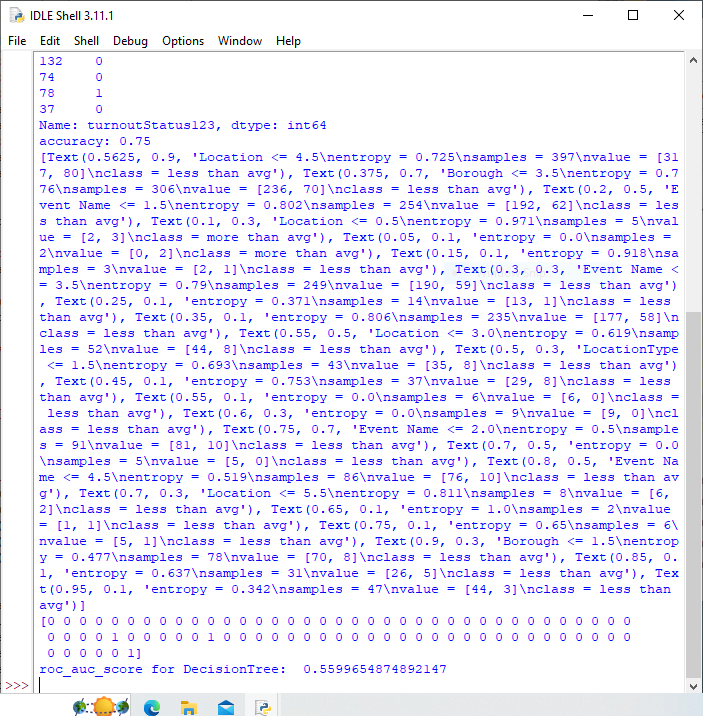


**6.6 MODEL VALIDATION**

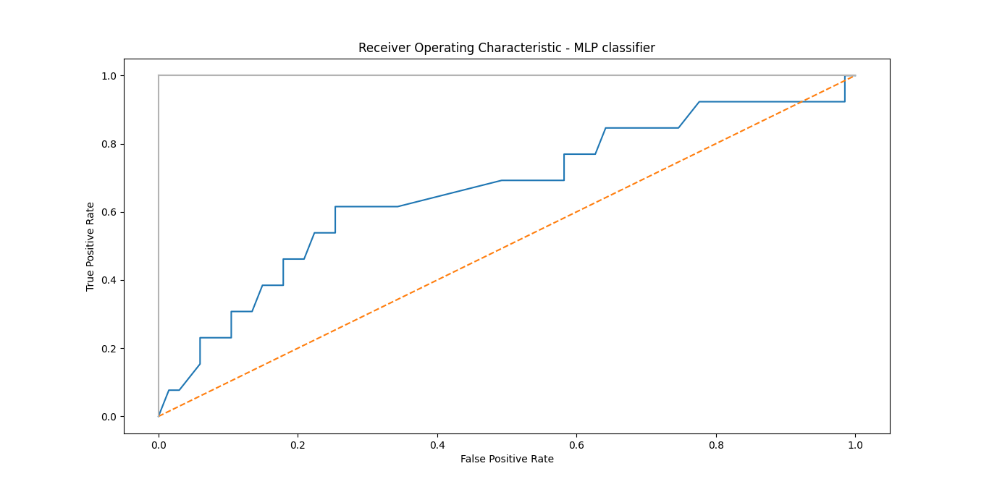
KNN



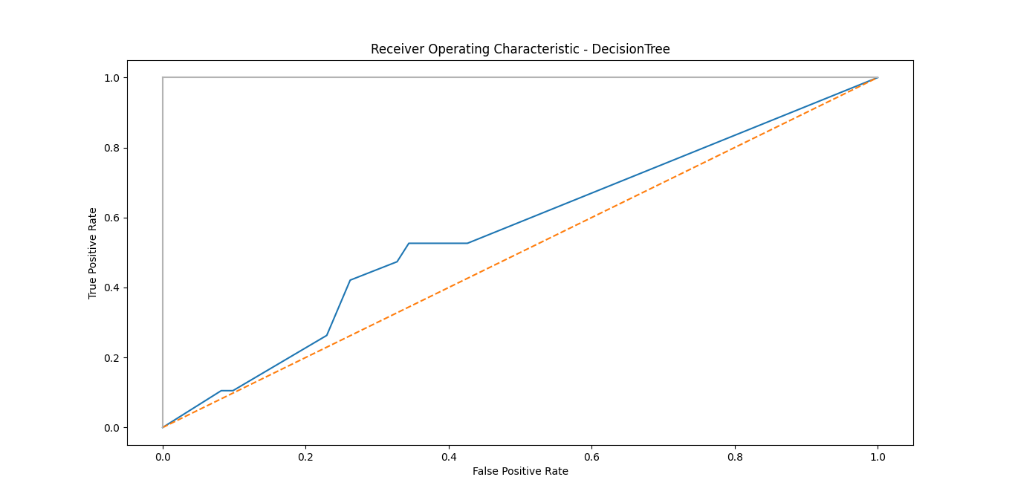
MLP



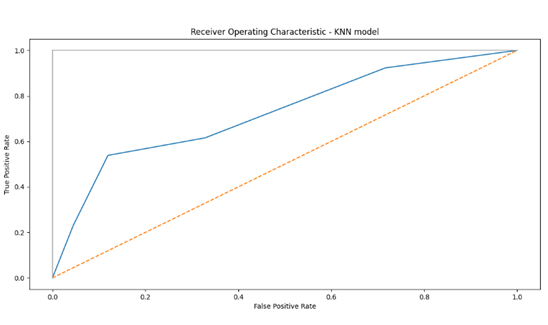
Decision Tree



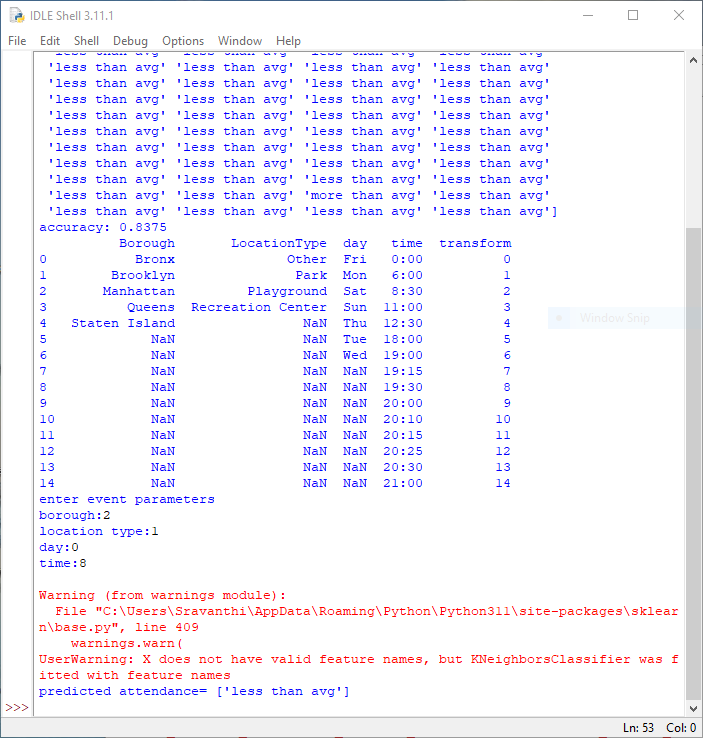
**Fig. 6.6.1 ROC-MLP Classifier**



**Fig. 6.6.2 ROC-Decision Tree**

****

**Fig. 6.6.3 ROC-KNN Model**



**6.7 DATA PRODUCT**

**CONCLUSION**

1. KNN (k-nearest neighbors), MLP (multi-layer perceptron), and Decision Tree classification models were fitted for this dataset and their respective accuracies, ROC curves, and AUCs were compared to finalize a predictive model.

2. After comparing the 3 models, accuracy of KNN(0.835) & MLP(0.835) models was better than the accuracy of decision tree classifier(0.75) and the AUC of KNN(0.725) was higher than MLP(0.662), hence the KNN model was finalized as the predictive model for this project.

3. Now parameter related values for borough, location type, day, and time of the event to be organized can be entered to predict if the attendance to this event will be less than or more than average (74) using the KNN model.

**REFERENCES**

https://www.geeksforgeeks.org/update-column-value-of-csv-in-python/

Dataset:

https://data.world/city-of-ny/6v4b-5gp4

Exploratory data analysis:

https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/

ROC:

https://www.statology.org/plot-multiple-roc-curves-python/

https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-algorithm-using-python-and-sci

Average attendance calculation :

<https://eventkickstart.com/average-event-cost>