import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

df=pd.read\_csv("traindata.csv")

df.head(5)

print(df.head(5))

df.shape

df.isnull().sum()

#print(df.isnull)

#removing seats

df['Seats'].mode()

df['Seats'].fillna(value=5.0,inplace=True)

df.info()

#removing kmpl and km/kg from mileage column

df['Mileage'] = df['Mileage'].apply(lambda x: str(x).replace('kmpl', '') if 'kmpl' in str(x) else str(x))

df['Mileage'] = df['Mileage'].apply(lambda x:str(x).replace('km/kg', '') if 'km/kg' in str(x) else str(x))

#removing CC from engine column

df['Engine'] = df['Engine'].apply(lambda x: str(x).replace('CC', '') if 'CC' in str(x) else str(x))

#removing bhp from power column

df['Power'] = df['Power'].apply(lambda x: str(x).replace('bhp', '') if 'bhp' in str(x) else str(x))

df['Mileage'] = pd.to\_numeric(df['Mileage'], errors='coerce')

df['Engine'] = pd.to\_numeric(df['Engine'], errors='coerce')

df['Power'] = pd.to\_numeric(df['Power'], errors='coerce')

df['Mileage'].mode()

df['Mileage'].fillna(value=17.0,inplace=True)

df['Engine'].mode()

df['Engine'].fillna(value=1197.0,inplace=True)

df['Power'].mode

df['Power'].fillna(value=74,inplace=True)

df.isnull().sum()

#print(df.isnull().sum())

#Since new price column has so many values will not use

df['Name'].nunique()

#print(df['Name'].nunique())

#The Name column has so many values so we will separate the brand names from the column and create a new column Brand\_Name.

df['Brand\_Name'] = df['Name'].str.split(' ').str[0]

df.groupby('Brand\_Name').nunique()

df['Brand\_Name'].unique()

#print(df['Brand\_Name'].unique())

#dropping the Name ,Location and new\_price column

df1\_map=df.drop(["Name","Location","New\_Price"],axis='columns')

df1\_map.head(5)

#print(df1\_map.head(5))

#new data frame

#heat map to find no null values in dataset

#sns.heatmap(df1\_map.isnull())

#plt.show()

plt.xlabel("Brand\_Name")

plt.ylabel("Count of car")

#df1\_map['Brand\_Name'].value\_counts().plot(kind='bar',title='Brand vs Car count',color='#C03928')

#plt.grid(color='black',linestyle='-.',linewidth=0.7)

#plt.show()

#The abv graph shows people buy Maruti and Hyundai car more than other brands

plt.xlabel("Year")

plt.ylabel("Count of car")

#df1\_map['Year'].value\_counts().plot(kind='bar',title='Year vs car count',color='#0E6251')

#plt.grid(color='black', linestyle='-.', linewidth=0.7)

#plt.show()

#Abv graph shows maximum number of cars in the data frame is between 2010 to 2017

#fuel-type

plt.xlabel("Fuel\_Type")

plt.ylabel("Count of car")

#df1\_map['Fuel\_Type'].value\_counts().plot(kind='bar',title='Fuel\_Type vs car count',color='black')

#plt.grid(linestyle='-.')

#plt.show()

#Transmission

plt.xlabel("Transmission")

plt.ylabel("Count of car")

#df1\_map['Transmission'].value\_counts().plot(kind='bar',title='Transmission vs car count',color='#C0392B')

#plt.grid(linestyle='-.')

#plt.show()

#owner type

plt.xlabel("Owner\_Type")

plt.ylabel("Count of car")

#df1\_map['Owner\_Type'].value\_counts().plot(kind='bar',title='Owner\_Type vs car count',color='blue')

#plt.grid(linestyle='-.')

#plt.show()

#seats

plt.xlabel("No of seats")

plt.ylabel("Count of car")

#df1\_map['Seats'].value\_counts().plot(kind='bar',title='Number of seats vs car count',color='cyan')

#plt.grid(linestyle='-.')

#plt.show()

#CONCLUSION 1.Max cars are of petrol 2.Manual cars are more 3.First hand cars are max 4.Cars with 5 seater are dominant

#BrandVsPrice=pd.DataFrame(df1\_map.groupby('Brand\_Name')['Price'].mean())

#BrandVsPrice.plot.bar(color='tomato',figsize=(11,5))

#plt.grid(linestyle='-.')

#plt.show()

#abv graph show Lamborghini is the most expensive car

#year vs price

plt.title("Year vs Price")

plt.xlabel("Year")

plt.ylabel("Price")

#plt.scatter(df1\_map.Year,df1\_map.Price)

#plt.show()

#fuel type vs price

plt.title("Fuel\_Type vs Price")

plt.xlabel("Fuel\_Type")

plt.ylabel("Price")

#plt.scatter(df1\_map.Fuel\_Type,df1\_map.Price)

#plt.show()

#transmission vs price

plt.title("Transmission vs Price")

plt.xlabel("Transmission")

plt.ylabel("Price")

#plt.scatter(df1\_map.Transmission,df1\_map.Price)

#plt.show()

#owner type vs price

plt.title("Owner\_Type vs Price")

plt.xlabel("Owner")

plt.ylabel("Price")

#plt.scatter(df1\_map.Owner\_Type,df1\_map.Price)

#plt.show()

#Cars ranging between the years 2012 to 2020 cost more.

#Petrol and diesel cars are costly.

#Automatic cars cost more

#First hand cars are costly

plt.title("Kilometers Driven vs Price")

plt.xlabel("Kilometers Driven")

plt.ylabel("Price")

#plt.scatter(df1\_map.Kilometers\_Driven,df1\_map.Price)

#plt.show()

#one of the cars has km drove more than 6500000, this is an outliner and we need to remove

#removing outlier

df1\_map.drop(df1\_map[df1\_map['Kilometers\_Driven'] >= 6500000].index, axis=0, inplace=True)

#mileage vs price

plt.title("Mileage vs Price")

plt.xlabel("Mileage")

plt.ylabel("Price")

#plt.scatter(df1\_map.Mileage,df1\_map.Price)

#Seats vs price

plt.title("Seats vs Price")

plt.xlabel("Seats")

plt.ylabel("Price")

#plt.scatter(df1\_map.Seats,df1\_map.Price)

#Some rows have zero values in mileage and seats column

df1\_map.isin([0]).sum()

#print(df1\_map.isin([0]).sum())

#Dropping 1 row from Seats column with zero value

df1\_map.drop(df1\_map[df1\_map['Seats']==0].index,axis=0,inplace=True)

#Cant drop rows with zero value in the ,ileage column sice we lose 68 rows, so we replace with dummy values

#we have already calculated the mode of milage column for filling

#null values which is 17.0

df1\_map["Mileage"].replace({0.0:17.0 },inplace=True)

df1\_map.isin([0]).sum()

#print(df1\_map.isin([0]).sum())

#Machine Learning algorithms work with a numeric value.

#creating a new dataframe

df2\_n = df1\_map.copy()

#Fuel type, Transmission, Owner type,and Brand Name are categorical columns

from sklearn.preprocessing import LabelEncoder

le\_Fuel\_Type=LabelEncoder()

le\_Transmission=LabelEncoder()

le\_Owner\_Type=LabelEncoder()

le\_Brand\_Name=LabelEncoder()

df2\_n['Fuel\_Type\_n']= le\_Fuel\_Type.fit\_transform(df2\_n['Fuel\_Type'])

df2\_n['Transmission\_n']=le\_Transmission.fit\_transform(df2\_n['Transmission'])

df2\_n['Owner\_Type\_n']=le\_Owner\_Type.fit\_transform(df2\_n['Owner\_Type'])

df2\_n['Brand\_Name\_n']=le\_Brand\_Name.fit\_transform(df2\_n['Brand\_Name'])

df2\_n.head(1)

#print(df2\_n.head(5))

#4 columns are created with numeric values

#Dropping columns with data type object

df2\_n=df2\_n.drop(["Fuel\_Type","Transmission","Owner\_Type","Brand\_Name"],axis='columns')

df2\_n.head(5)

#print(df2\_n.head(5))

#Shuffling columns as per our needs

df2\_n=df2\_n[['Brand\_Name\_n','Year','Kilometers\_Driven','Fuel\_Type\_n','Transmission\_n','Owner\_Type\_n','Mileage','Engine','Power','Seats','Price']]

df2\_n.head(1)

#print(df2\_n.head(1))

#Correlation matrix

corrMatrix = df2\_n.corr()

plt.figure(figsize=(10,7))

#sns.heatmap(corrMatrix, annot=True,cmap= 'coolwarm', linewidths=3, linecolor='black')

#plt.show()

#Creating 2 new data frames

df3\_inputs=df2\_n.drop(["Price"],axis='columns')

df3\_target=df2\_n['Price']

df3\_inputs.head(5)

#print(df3\_inputs.head(5))

df3\_target.head(5)

#print(df3\_target.head(5))

#‘df3\_inputs’ data frame has input features and ‘df3\_target’ data frame has the target value that we need to predict i.e price.

#least and imp feature used for predictiion

from sklearn.ensemble import ExtraTreesRegressor

import matplotlib.pyplot as plt

model = ExtraTreesRegressor()

model.fit(df3\_inputs,df3\_target)

#use inbuilt class feature\_importances of ExtraTreeRegressor

#plot graph of feature importances for better visualization

feat\_importances = pd.Series(model.feature\_importances\_, index=df3\_inputs.columns)

plt.figure(figsize=(11,5))

plt.xlabel("Value")

plt.ylabel("Features")

plt.title("Features vs Importance")

plt.grid()

#feat\_importances.nlargest(10).plot(kind='barh',color='#D98880')##45B39D

#plt.grid(color='black', linestyle='-.', linewidth=0.7)

#plt.show()

#Transmission\_n and Owner\_Type\_n are the most and least important features for predicting the price of a used car.

#Applying different modes on the data

from sklearn import linear\_model

from sklearn.linear\_model import Lasso

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

#Splitting data as test and rain

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

X\_train,X\_test,y\_train,y\_test=train\_test\_split(df3\_inputs,df3\_target,test\_size=0.2,random\_state=10)

len(X\_train)

#print(len(X\_train))

len(X\_test)

#print(X\_test)

#Training

Model\_RandomForest = RandomForestRegressor(max\_features='sqrt',bootstrap='True')

Model\_RandomForest.fit(X\_train,y\_train)

RandomForestRegressor(bootstrap='True', ccp\_alpha=0.0,criterion='mse',

max\_depth=None, max\_features='sqrt', max\_leaf\_nodes=None,

max\_samples=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1,

min\_samples\_split=2,min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_jobs=None, oob\_score=False,

random\_state=None,verbose=0, warm\_start=False)

Model\_RandomForest.score(X\_test,y\_test)

#print(Model\_RandomForest.score(X\_test,y\_test))

#importing model

#pickel method

import pickle

#wrinting the model in a file

pickle.dump(Model\_RandomForest,open('rmodel.pk1','wb'))

#reading the file

rmodel=pickle.load(open('rmodel.pk1','rb'))

#test model

rmodel.predict([[17,2015,100000,15,1000,40,5.0,3,1,0]])

print(rmodel.predict([[17,2015,100000,15,1000,40,5.0,3,1,0]]))