import pandas as pd

import numpy as np

SampleCarDataset = pd.read\_csv("21301576\_MD Ikramul Kayes\_CSE422\_11\_Assignment04\_Summer2024.csv") #Here I am loading the dataset into the variable named SampleCarDataset

SampleCarDataset.head(10) #shows the first 10 rows

SampleCarDataset.tail() #seeing the bottom part of the dataset

print(SampleCarDataset.shape) #seeing the shape of the dataset

print(SampleCarDataset.columns) #seeing the features

print(SampleCarDataset.info()) #to get more insight of the dataset

print(SampleCarDataset.describe()) #see the contains statistics of each features

## 1) Handling NULL Value

SampleCarDataset.isnull().sum() #Checking if null values exists

SampleCarDataset = SampleCarDataset.dropna(subset=['tax']) #As we are seeing 9 null values in tax feature. So, we will drop those 9 rows which contains the null value for tax column. As our dataset size is close to 18 thousand so dropping these 9 rows will have no effect on the learning

SampleCarDataset.isnull().sum() #Checking if any null value still exists

## 2) Duplicate Value check and Remove

duplicate\_rows = SampleCarDataset[SampleCarDataset.duplicated()] #taking the rows which are duplicate but avoids the first occured row

print(duplicate\_rows) #printing the duplicate rows

duplicate\_count = SampleCarDataset.duplicated().sum() #getting the duplicate rows

print(duplicate\_count)

SampleCarDataset = SampleCarDataset.drop\_duplicates() #Here we are dropping the rows which are duplicate. As duplicates value leads to overfitting as the model may memorize the repeating instances instead of finding pattern between different datas. Also, it may create biasness in model by observing the duplicate data multiple times

duplicate\_count = SampleCarDataset.duplicated().sum() # Checking if any duplicate still exists

print(duplicate\_count)

## 3) Handling Categorical Datas

SampleCarDataset['model'].unique() #Here we are seeing unique categorical datas in the column model

SampleCarDataset['transmission'].unique() #Here we are seeing unique categorical datas in the column transmission

SampleCarDataset['fuelType'].unique() #Here we are seeing unique categorical datas in the column fuelType

#As model do not understand categorical datas we need to convert them in a way so that model can understand. For this we will do onehot encoding which will create different rows for each unique categorical data in that column

SampleCarDataset = pd.get\_dummies(SampleCarDataset, columns=['transmission'], drop\_first=True) # Here 2 new column Manual and Semi Auto will be created replacing the transmission column. As both of them being false represents that the car is Automatic.

SampleCarDataset.head(10) #shows the first 10 rows

SampleCarDataset = pd.get\_dummies(SampleCarDataset, columns=['fuelType'], drop\_first=True) # Same is here we are dropping the fuelType column and creating rows for each unqiue categorical datas

SampleCarDataset.head(10) #shows the first 10 rows after the one hot encoding

SampleCarDataset = pd.get\_dummies(SampleCarDataset, columns=['model'], drop\_first=True) # Same is here we are dropping the model column and creating rows for each unqiue categorical datas

SampleCarDataset.head(10) #shows the first 10 rows after the one hot encoding of model column

## 4) Feature scaling

# As High variance feature leads to model give more weights or priorty to them and ignores low variance features. This may lead to model not gaining trained properly

# This is why we need to minimize the variance in our dataset for this we do feature scaling

# Separating features and target variable

X = SampleCarDataset.drop(columns=['price']) # Our features

y = SampleCarDataset['price'] # Our target

print(X.describe())

from sklearn.preprocessing import StandardScaler # Importing StandardScaler

import pandas as pd

# Here we are seeing the mileage and tax columns have the high standard deviation which means that both of these features have high variance data.

# Selecting the 'mileage' and 'tax' columns to scale

columns\_to\_scale = ['mileage', 'tax']

# Initializing the StandardScaler

scaler = StandardScaler() #As our features in our dataset are of different units so it is best for us to choose StandardScaler for feature scaling

# Lets Apply the scaler to the selected columns

X[columns\_to\_scale] = scaler.fit\_transform(X[columns\_to\_scale])

print(X[columns\_to\_scale].describe())

X.describe()

final\_data = pd.concat([X, y], axis=1)

final\_data.head(100)

## 5) Corelation Value

# We need to check which features corelate with each other. As Corelated features are redducndent and bear the same pattern. This is why we need to find the corelation of the database.

X\_corr = X.corr()

X\_corr

import seaborn as sns

import matplotlib.pyplot as plt

# Create a larger figure

plt.figure(figsize=(12, 8)) # Adjust the width and height as needed

# Create the heatmap

sns.heatmap(X\_corr, cmap='YlGnBu')

# Display the heatmap

plt.show()

#After observing the heatmap we can say that,the year and mileage are highly corelated.

features\_to\_drop = ['year']

X\_reduced = X.drop(columns=features\_to\_drop)

X\_reduced.head()

X\_reduced.columns