Supervised Learning- Regression Car Pricing</h2> Problem Description: A Chinese automobile company aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts. # importing necessary libraries In [13]: import pandas as pd import numpy as np from sklearn import preprocessing from sklearn import linear_model import matplotlib.pyplot as plt import seaborn as sns from sklearn.preprocessing import LabelEncoder In [2]: # loading the dataset df=pd.read_csv("C://Users//EliteBook//Desktop//ds//project//CarPr car_ID symboling CarName fueltype aspiration doornumber carbody Out[2]: alfa-romero 0 1 3 gas std two convertible giulia alfa-romero 2 3 std convertible gas two stelvio alfa-romero hatchback 2 3 1 std two gas Quadrifoglio 4 2 audi 100 ls sedan std four gas 2 4 5 audi 100ls std four sedan gas volvo 145e 200 201 -1 four std sedan gas (sw) 202 volvo 144ea 201 -1 gas turbo four sedan 202 203 volvo 244dl gas std four sedan 203 volvo 246 204 diesel turbo four sedan -1 204 205 volvo 264gl turbo four -1 gas sedan 205 rows × 26 columns Exploratory Analysis</h2> In [4]: df.shape (205, 26)Out[4]: In [5]: df.head() aspiration carbody dri Out[5]: symboling CarName fueltype doornumber car_ID alfa-romero 0 1 std convertible two gas giulia alfa-romero convertible gas two stelvio alfa-romero 2 3 std hatchback two gas Quadrifoglio 3 audi 100 ls gas std four sedan 4 5 audi 100ls gas std four sedan 5 rows × 26 columns df.info() In [6]: <class 'pandas.core.frame.DataFrame'> RangeIndex: 205 entries, 0 to 204 Data columns (total 26 columns): # Column Non-Null Count Dtype ---------205 non-null int64 0 car_ID 1 symboling 205 non-null int64 2 CarName 205 non-null object 3 fueltype 205 non-null object 4 aspiration 205 non-null object 5 doornumber 205 non-null object 6 205 non-null carbody object 7 drivewheel 205 non-null object 205 non-null 8 enginelocation object 9 wheelbase 205 non-null float64 carlength 10 205 non-null float64 205 non-null float64 11 carwidth 12 205 non-null float64 carheight 205 non-null 13 curbweight int64 205 non-null 14 object enginetype 205 non-null 15 cylindernumber object 16 enginesize 205 non-null int64 17 205 non-null fuelsystem object 18 boreratio 205 non-null float64 19 stroke 205 non-null float64 compressionratio 205 non-null 20 float64 21 horsepower 205 non-null int64 205 non-null int64 peakrpm citympg 205 non-null int64 highwaympg 205 non-null int64 price 205 non-null dtypes: float64(8), int64(8), object(10) memory usage: 41.8+ KB In [7]: df.columns Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiratio Out[7]: 'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight', 'engin etype', 'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'h ighwaympg', 'price'], dtype='object') # Count missing values in each column In [8]: df.isnull().sum() car_ID 0 Out[8]: symboling 0 CarName 0 0 fueltype aspiration 0 doornumber 0 0 carbody drivewheel 0 enginelocation 0 wheelbase carlength 0 carwidth 0 carheight 0 curbweight 0 0 enginetype cylindernumber 0 enginesize 0 fuelsystem 0 boreratio 0 stroke 0 compressionratio 0 horsepower 0 peakrpm 0 citympg 0 highwaympg 0 price 0 dtype: int64 In [11]: # Convert categorical variables using Label Encoding label_encoder = LabelEncoder() categorical_cols = ['fueltype', 'aspiration', 'doornumber', 'carbo for col in categorical_cols: df[col] = label_encoder.fit_transform(df[col]) In [12]: # Display the cleaned dataset df.head() CarName fueltype aspiration doornumber carbody drive Out[12]: car_ID symboling alfa-romero 0 1 1 0 1 0 giulia alfa-romero 2 1 1 1 0 stelvio alfa-romero 2 2 3 1 0 1 Quadrifoglio 3 audi 100 ls 4 5 audi 100ls 1 0 0 3 2 5 rows × 26 columns # Split the 'CarName' column into 'CarName' and 'CompanyName' In [25]: df[['CarName', 'CompanyName']] = df['CarName'].str.split(' ', 1, df.head() In [26]: car_ID symboling CarName fueltype aspiration doornumber carbody drivew Out[26]: alfa-0 1 0 0 3 1 1 romero alfa-2 0 1 3 1 romero alfa-2 3 0 2 1 1 1 romero 3 4 2 audi 1 0 0 3 5 audi 1 0 3 5 rows × 27 columns for col in df.describe(include = 'object').columns: In [27]: print(col) print(df[col].unique()) CarName ['alfa-romero' 'audi' 'bmw' 'chevrolet' 'dodge' 'honda' 'isuzu' jaguar' 'maxda' 'mazda' 'buick' 'mercury' 'mitsubishi' 'Nissan' 'nissan' 'peugeot' 'plymouth' 'porsche' 'porcshce' 'renault' 'saab' 'suba ru' 'toyota' 'toyouta' 'vokswagen' 'volkswagen' 'vw' 'volvo'] CompanyName ['giulia' 'stelvio' 'Quadrifoglio' '100 ls' '100ls' 'fox' '5000' '4000' '5000s (diesel)' '320i' 'x1' 'x3' 'z4' 'x4' 'x5' 'impala' 'monte carlo' 'vega 2300' 'rampage' 'challenger se' 'd200' 'monaco (sw)' 'colt hardtop' 'colt (sw)' 'coronet custom' 'dart custom' 'coronet custom (sw)' 'civic' 'civic cvcc' 'accord cvcc' 'accord lx' 'civic 1500 gl' 'accord' 'civic 1300' 'prelude' 'civic (auto)' 'MU-X' 'D-Max ' 'D-Max V-C 'xj' 'xf' 'xk' 'rx3' 'glc deluxe' 'rx2 coupe' 'rx-4' '626' 'glc' 'rx-7 gs' 'glc 4' 'glc custom l' 'glc custom' 'electra 225 custo m' 'century luxus (sw)' 'century' 'skyhawk' 'opel isuzu deluxe' 'sk ylark' 'century special' 'regal sport coupe (turbo)' 'cougar' 'mirage' 'lancer' 'outlander' 'g4' 'mirage g4' 'montero' 'pajero' 'versa' 'gt-r' 'rogue' 'latio' 'titan' 'leaf' 'juke' 'note' 'clipper' 'nv200' 'dayz' 'f uga' 'otti' 'teana' 'kicks' '504' '304' '504 (sw)' '604sl' '505s turb o diesel' 'fury iii' 'cricket' 'satellite custom (sw)' 'fury gran sedan' 'valiant' 'duster' 'macan' 'panamera' 'cayenne' 'boxter' '12tl' '5 gtl' '9 9e' '99le' '99gle' None 'dl' 'brz' 'baja' 'r1' 'r2' 'trezia' 'tribec a' 'corona mark ii' 'corona' 'corolla 1200' 'corona hardtop' 'corolla 1600 (sw)' 'carina' 'mark ii' 'corolla' 'corolla liftba ck' 'celica gt liftback' 'corolla tercel' 'corona liftback' 'starle t' 'tercel' 'cressida' 'celica gt' 'rabbit' '1131 deluxe sedan' 'mo del 111' 'type 3' '411 (sw)' 'super beetle' 'dasher' 'rabbit custom' '145 e (sw)' '144ea' '244dl' '245' '264gl' 'diesel' '246'] len(df.CarName.unique()) In [15]: 147 Out[15]: df.describe() In [16]: symboling doornumber Out[16]: car_ID fueltype aspiration carbody 205.000000 205.000000 205.000000 count 205.000000 205.000000 205.000000 103.000000 0.834146 0.439024 mean 0.902439 0.180488 2.614634 0.859081 std 59.322565 1.245307 0.297446 0.385535 0.497483 min 1.000000 -2.000000 0.000000 0.000000 0.000000 0.000000 2.000000 25% 52.000000 0.000000 1.000000 0.000000 0.000000 50% 103.000000 1.000000 1.000000 0.000000 0.000000 3.000000 75% 154.000000 2.000000 1.000000 0.000000 1.000000 3.000000 max 205.000000 3.000000 1.000000 1.000000 1.000000 4.000000 8 rows × 25 columns Explore relationships between variables using visualizations. # Correlation matrix corr_matrix = df.corr() In [18]: # Heatmap of correlations plt.figure(figsize=(12, 8)) sns.heatmap(corr_matrix, annot=True, cmap="coolwarm") plt.title("Correlation Heatmap") Correlation Heatmap 1.00 car_ID - 1 0.150.130.0680.130.098.0510.0510.130.170.0520.260.0740.076.040.0340.2 0.260.160.150.0150.20.016.0110.11 symboling -0.15 1 0.190.060.66 -0.60.04 2.21 0.5 -0.360.220.5 -0.230.05 0.2 -0.1 0.09 10.16.00 8 7.1 0.07 10.27 0.03 603 50.08 fueltype -0.13<mark>0.19 1 -0.40.19</mark>0.150.130.040.310.210.230.280.2**2**0.0830.11-0.010.042.0540.240.980.160.480.260.190.11 0.75 aspiration e.0680.06-0.4 1 0.032068.066.05 0.260.23 0.30.08 0.32 -0.1-0.130.110.290.210.22 0.3 0.240.18-0.2-0.250.18 doornumber -0.190.66 0.190.03 1 -0.60.099.140.45-0.4-0.210.55 0.20.0620.190.020.0120.110.10110.110.130.25 0.19.036 0.32 carbody e0.098 0.6 0.190.06 0.68 1 0.160.28 0.4 0.330.130.57 0.130.030.048.0 78.065.0190.019.140.150.10.032.0072084 drivewheel 0.050.0420.10.066.0990.16 1 0.150.460.490.470.020.550.120.220.520.420.480.0720.130.520.039.450.450.56 enginelocation 0.0510.210.040.0510.140.280.15 1 0.190.050.0520.110.050.110.14 0.2 0.110.190.140.020.32 0.2 0.150.1 0.32 0.50 0.78<mark>0.140.18</mark>0.570.380.49<mark>0.160.250.35</mark>0.360.470 0.25 enginetype -0.07**5**.0**5**.0830.10.062.03-0.12<mark>0.11</mark>0.140.1 D.0120.1-0.055<mark>11</mark>0.249.04 D.092.0250.1-0.0720.00.00566.085.0768.049 cylindernumber -0.04<mark>10.2 0.11</mark>-0.130.150.04<mark>8.220.1</mark>40.180.110.190.280.04<mark>70.24 1 0.086.012</mark>0.0320.030.050.0650.120.220.130.086.028 enginesize -0.0340.110.07<mark>0.11</mark>0.020.07 0.52 0.2 0 .570.680.74<mark>0.067</mark>0.85<mark>0.0410.08811 | 0.510.58 0.20.024</mark>0.81<mark>0.24</mark>0.650.680.87 -0.25 fuelsystem - 0.20.09@.0420.290.016.06 5.420.110.380.560.520.0170.610.0920120.51 1 0.480.0880.1 0.60.0140.670.650.53 boreratio -0.260.130.054.21-0.120.0110.480.190.490.610.560.170.650.0249.033.580.48 1 0.0560050.570.25 stroke -0.16.0080.240.220.01-0.015.0720.140.160.130.180.055.17-0.140.050.20.080.050110.190.080.068.040.040.040.079 -0.50compressionratio -0.150.1<mark>20.98 0.3-</mark>0.180.140.130.020.250.160.180.260.150.070.0650290.0.0050.19 1 -0.2-0.440.320.270.068 horsepower -0.01050710.160.240.130.150.520.320.350.550.640.110.750.010.120.810.660.570.0810.2 1 0.13-0.8-0.770.81 peakrpm - 0.2 <mark>0.270.48</mark>0.18<mark>0.25</mark>0.110.03<mark>90.2</mark>-0.360.290.220.320.207.005<mark>6.22</mark>0.24.0140.250.06**8**0.44<mark>0.13 1 -</mark>0.1-0.056.085 -0.75highwaympg 0.010.0350.190.25.036007<mark>0.45-0.1-0.54-0.7-0.680.11-0.80.076.08-0.680.650.5-0.04</mark>0.27-0.7-0.05 price -0.110.080.110.180.030.084.58<mark>0.32</mark>0.580.680.760.12<mark>0.84</mark>0.048.02<mark>0.870.530.54</mark>0.078.0660.810.0850.6 # Scatter plots In [28]: sns.pairplot(df, x_vars=['horsepower', 'curbweight', 'enginesize' plt.show() 40000 20000 10000 2000 4000 300 curbweight enginesize Analyze categorical variables and their distribution. In [29]: # Countplot of a categorical variable plt.figure(figsize=(10, 6)) sns.countplot(df['fueltype']) plt.title("Distribution of Fuel Type") plt.show() C:\Users\EliteBook\anaconda3\lib\site-packages\seaborn_decorator s.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument wil 1 be `data`, and passing other arguments without an explicit keyw ord will result in an error or misinterpretation. warnings.warn(Distribution of Fuel Type 175 150 125 100 75 25 0 fueltype Based on our analysis, we found the following distribution of fuel types in the dataset: Gas (Petrol) Cars: [belowe 25] Diesel Cars: [Above 175] Data Preprocessing</h2> In [30]: # There are non numeric values for some columns. They should be co le = preprocessing.LabelEncoder() le.fit(df.fueltype) df.fueltype = le.transform(df.fueltype) le.fit(df.aspiration) In [31]: df.aspiration = le.transform(df.aspiration) In [32]: le.fit(df.doornumber) df.doornumber = le.transform(df.doornumber) le.fit(df.carbody) In [33]: df.carbody = le.transform(df.carbody) le.fit(df.drivewheel) In [34]: df.drivewheel = le.transform(df.drivewheel) le.fit(df.enginelocation) In [35]: df.enginelocation = le.transform(df.enginelocation) le.fit(df.enginetype) In [37]: df.enginetype = le.transform(df.enginetype) le.fit(df.cylindernumber) In [38]: df.cylindernumber = le.transform(df.cylindernumber) le.fit(df.fuelsystem) In [39]: df.fuelsystem = le.transform(df.fuelsystem) df.head() In [42]: car_ID symboling CarName fueltype aspiration doornumber Out[42]: carbody drivew alfa-0 1 3 1 0 1 0 romero alfa-1 2 3 romero alfa-2 3 1 1 0 1 2 romero 3 4 2 audi 0 3 4 5 audi 1 0 0 3 5 rows × 27 columns df[df.duplicated()] In [41]: car_ID symboling CarName fueltype aspiration doornumber carbody drivewh 0 rows × 27 columns Modelling using Linear Regression</h2> reg = linear_model.LinearRegression() In [43]: ind_var = df[['fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'u'carlength', 'carwidth', 'carheight', 'curbweight', 'engine 'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'h dep_var = df.price reg.fit(ind_var,dep_var) LinearRegression() Out[43]: In [44]: reg.coef_ array([-9.21677014e+03, 2.42352286e+02, -5.13930613e+02, -8.1913 Out[44]: 9857e+02, 1.16616708e+03, 1.03974427e+04, 1.10593867e+02, -2.9110 2195e+01, 7.12602036e+02, 1.84232716e+02, 2.35662037e+00, 1.9094 8576e+02, 1.28166071e+02, 1.01622195e+02, -1.48294905e+02, -2.7445 0799e+03, -2.82460445e+03, -5.48092812e+02, 2.06295041e+01, 2.1044 0991e+00, -1.45461967e+02, 1.35183217e+02]) In [45]: reg.intercept_ -49313.25002503388 Out[45]: reg.predict([[1,0,1,0,2,0,88.6,168.8,64.1,48.8,2548,0,0,13,0,1,0, In [51]: C:\Users\EliteBook\anaconda3\lib\site-packages\sklearn\base.py:45 0: UserWarning: X does not have valid feature names, but LinearRe gression was fitted with feature names warnings.warn(array([12962.15203553]) Out[51]: In [49]: | df['price'][0] 13495.0 Out[49]: Summery</h2> The predicted price is similar to actual price. So the model works fine