

## Step 1 : Download the dataset

The screenshot shows the Kaggle website interface. The main heading is 'Automobile Dataset' by RAMAKRISHNAN SRINIVASAN, updated 6 years ago. It has 361 views and a 'Download (5 kB)' button. The dataset description states: 'Dataset consist of various characteristic of an auto'. Below this, there are tabs for 'Data Card', 'Code (172)', and 'Discussion (6)'. A section titled 'Dataset Notebooks' is visible, with a search bar and filters. The browser's address bar shows the URL: <https://www.kaggle.com/datasets/toranky/automobile-dataset/download?datasetVersionNumber=2>.

## Step 2: import libraries and read the csv using pandas

The screenshot shows an Anaconda Jupyter Notebook environment. The code in the cell is as follows:

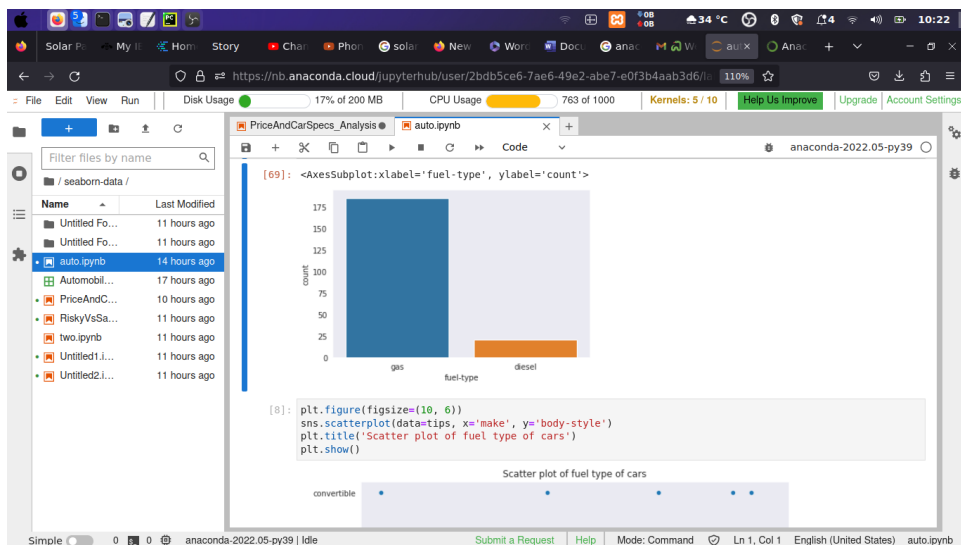
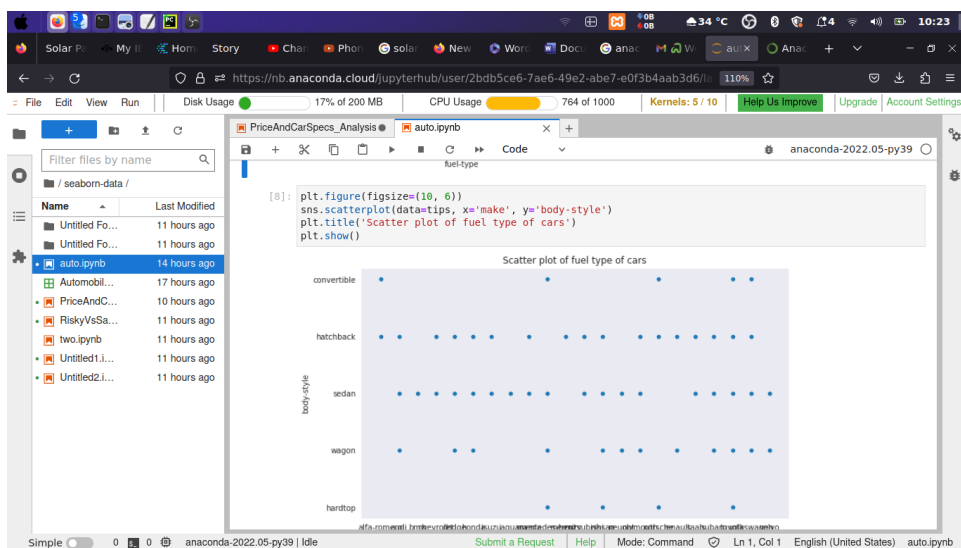
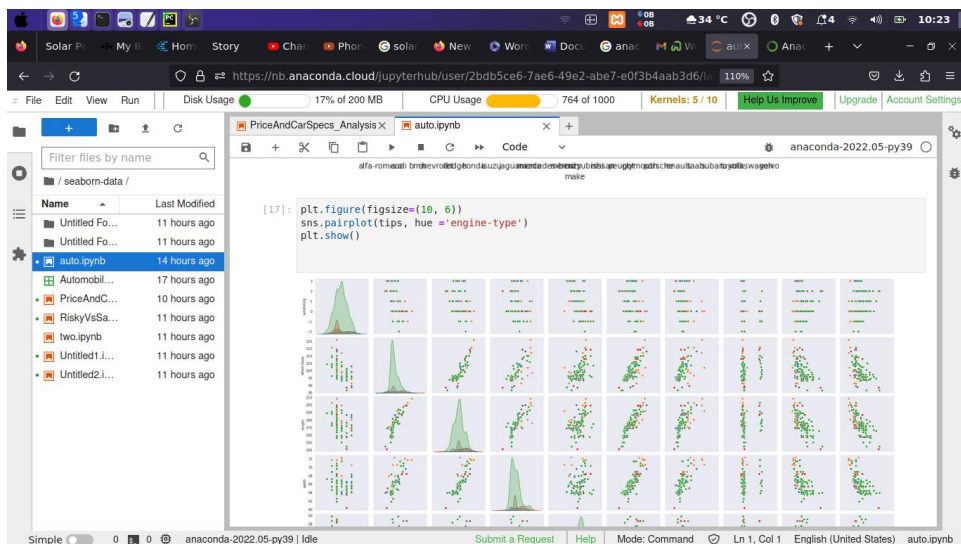
```
[69]: import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
import pandas as pd

tips = sns.load_dataset('Automobile_data')
sns.set_style('dark')

sns.countplot(x='fuel-type', data=tips)
```

The output of the code is a bar chart showing the count of data points for each fuel type. The x-axis is labeled 'fuel-type' and the y-axis is labeled 'count'. The chart shows two bars: one for 'gas' (count ~175) and one for 'diesel' (count ~100).

## Step 3: perform Univariate, bivariate and Tri variate analysis



## Step 4: next Handling the missing value and splitting the data

The image displays two screenshots of a Jupyter Notebook interface, likely from Anaconda, showing the process of handling missing values and splitting data for an automobile price prediction model.

**Top Screenshot:** The notebook is titled "Untitled.ipynb" and shows the following code and text:

```
[ ]: X = encoded_data.drop('price', axis = 1)
     y = encoded_data['price']

[ ]: print(X.shape,y.shape)
```

split data into train and test set

```
[ ]: from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state = 15)

[ ]: print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

We are using 67% data points for training and 33% data points for test.

**Data Standardization**

Standardization data means bringing all attributes on same scale with mean = 0 and standard deviation = 1.

```
[ ]: scaler = StandardScaler()
     std_X_train = scaler.fit_transform(X_train)
     std_X_test = scaler.transform(X_test)
```

**Model deployment**

**Bottom Screenshot:** The notebook is titled "Untitled.ipynb" and shows the following code and text:

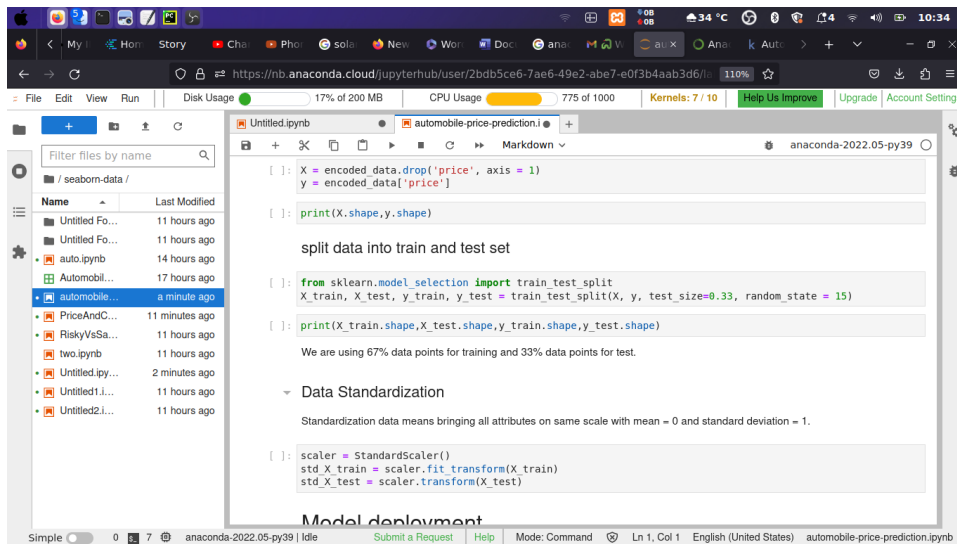
```
[ ]: missing_data = data.isnull()
     for column in missing_data.columns.values.tolist():
         print(column)
         print (missing_data[column].value_counts())
         print("")
```

Based on the summary above, following columns has missing values:

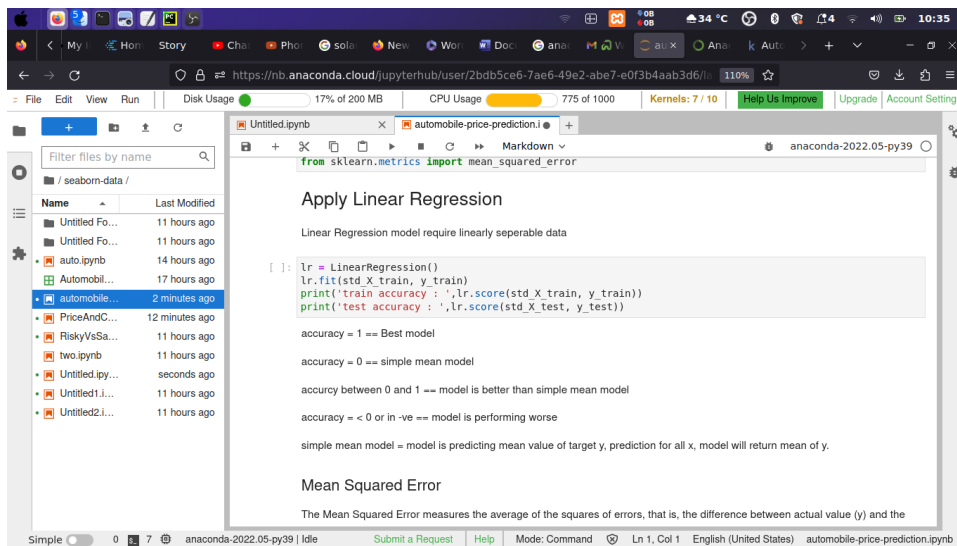
- 1."normalized-losses": 41 missing data
- 2."num-of-doors": 2 missing data
- 3."bore": 4 missing data
- 4."stroke": 4 missing data
- 5."horsepower": 2 missing data
- 6."peak-rpm": 2 missing data
- 7."price": 4 missing data

```
[ ]: data.info()
```

## Step 5 : Handling categorical variable and standard scaling



## Step 6: apply linear regression



For SVR model we got train accuracy = -0.126 and test accuracy = 0.00049 which is less than Ridge and Lasso regression.

### Compare all models

```
[ ]: # http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
from prettytable import ALL as ALL
table=PrettyTable(rules=ALL)
table.field_names = ["Sr.No.", "Model", "Train accuracy", "Test accuracy", "MSE"]
table.add_row([1, "Linear Regression", 0.974 , -2.005e+23, 5.57e+30 ])
table.add_row([2, "Ridge Regression", 0.972 , 0.883, 3259996.36 ])
table.add_row([3, "Lasso Regression", 0.974 , 0.822, 4948475.08 ])
table.add_row([4, "Polynomial SVR", -0.12 , 0.00049, 27794961.60 ])

print(table)
```

Comparing above models Ridge Regression got high accuracy.

### Deploy new model by dropping less important features

Now, we have got best model for price prediction. But here we have considered all attributes. For models like linear regression we should not consider attributes which are correlated to each other, numerical attributes which are non linearly related to our

```
lasso.fit(std_X_train, y_train)
print('train accuracy : ',lasso.score(std_X_train, y_train))
print('test accuracy : ',lasso.score(std_X_test, y_test))
```

### Mean Squared Error

```
[ ]: y_pred = lasso.predict(std_X_test)
print('MSE = ',mean_squared_error(y_pred, y_test))
```

### Apply support vector regressor

```
[ ]: poly = SVR(kernel = 'poly', degree = 3 ,C=1.0)
poly.fit(std_X_train, y_train)
print('train accuracy : ',poly.score(std_X_train, y_train))
print('test accuracy : ',poly.score(std_X_test, y_test))
```

### Mean Squared Error

```
[ ]: y_pred = poly.predict(std_X_test)
print('MSE = ',mean_squared_error(y_pred, y_test))
```

### Compare model after removing less important features

STEP 7 : Prediction based on above models

anaconda2022.05-py39 | Idle

Submit a Request Help Mode: Command Ln 1, Col 1 English (United States) automobile-price-prediction.ipynb

10:37

34 °C

110%

Help Us Improve Upgrade Account Settings

Kernels: 7 / 10

CPU Usage 777 of 1000

Disk Usage 17% of 200 MB

File Edit View Run

anaconda.cloud/jupyterhub/user/2bdb5ce6-7ae6-49e2-abe7-e0f3b4aab30

seaborn-data /

Name	Last Modified
Untitled Fo...	11 hours ago
Untitled Fo...	11 hours ago
auto.ipynb	14 hours ago
Automobil...	17 hours ago
automobile...	a minute ago
PriceAndC...	14 minutes ago
RiskyVsSa...	11 hours ago
two.ipynb	11 hours ago
Untitled.ipy...	2 minutes ago
Untitled1.i...	11 hours ago
Untitled2.i...	11 hours ago

```
[ ]: def predict_price(wheel_base, width,engine_size, bore,horsepower,highway_mpg,make,fuel,aspiration,
    body_style,drive_wheels,engine_location,num_of_cylinders,fuel_system):
    loc_index = np.where(X.columns==make)[0][0]
    loc_index = np.where(X.columns==fuel)[0][0]
    loc_index = np.where(X.columns==aspiration)[0][0]
    loc_index = np.where(X.columns==body_style)[0][0]
    loc_index = np.where(X.columns==drive_wheels)[0][0]
    loc_index = np.where(X.columns==engine_location)[0][0]
    loc_index = np.where(X.columns==num_of_cylinders)[0][0]
    loc_index = np.where(X.columns==fuel_system)[0][0]

    x = np.zeros(len(X.columns))
    x[0] = wheel_base
    x[1] = width
    x[2] = engine_size
    x[3] = bore
    x[4] = horsepower
    x[5] = highway_mpg
    if loc_index >= 0:
        x[loc_index] = 1

    return pipe.predict([x])[0]

[ ]: predict_price(90, 80,150, 3.50,120,25,'make_saab','fuel-type_gas','aspiration_std','body-style_sedan','d
    'num-of-cylinders_four','fuel-system_4bbl')
```