Project Report

Course Code: CSE 413

Course Title: Artificial Intelligence Lab

Semester: Spring-2021

Submitted to:

Mr. Afjal Hossan Sarower

Lecturer of

CSE

Department of CSE

Daffodil International University

Submitted by:

Humayun Kabir -- ID: 181-15-2045

Maruf Hasan – ID: 181-15-1927

Ekramul Haque – ID: 181-15-1817

Section: PC-F

Project Report

Date: 16 April, 2021

Project Title

Car Price Prediction

Objectives:

This project aims to predict the Price of a Used Car by taking its Company name, its Model name, Year of Purchase, and other parameters.

A car price prediction has been a high-interest research area, as it requires noticeable effort and knowledge of the field expert. A considerable number of distinct attributes are examined for reliable and accurate prediction. To build a model for predicting the price of used cars the applied three machine learning techniques are Artificial Neural Network and linear regression. Respective performances of different algorithms were then compared to find one that best suits the available data set. The final prediction model was integrated into the Java application. Furthermore, the model was evaluated using test data and an accuracy of 82% was obtained.

Content Features:

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle’s price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in the United States. Our results show that the Random Forest model and K-Means clustering with linear regression yield the best results, but are compute-heavy. Conventional linear regression also yielded satisfactory results, with the advantage of a significantly lower training time in comparison to the aforementioned methods.

Development Tools and Technology:

- Jupyter Notebook
- VS code
- numpy
- pandas
- scikit-learn

Which Algorithm Use:

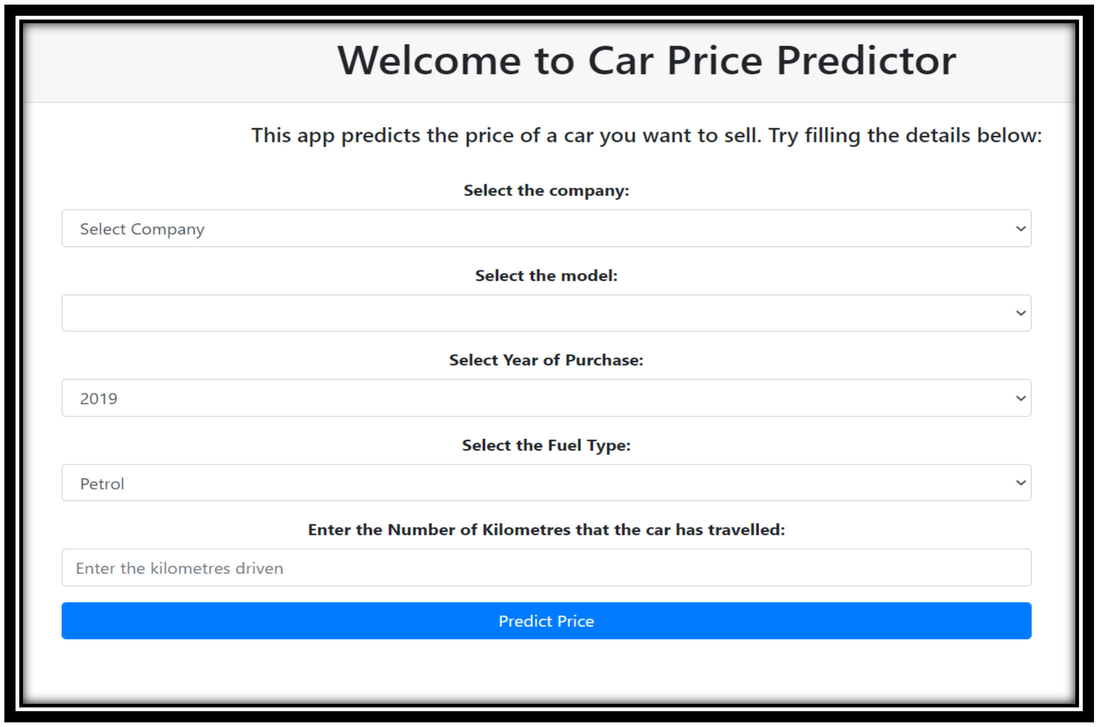
Linear regression:

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

User Benefits:

Actually, used car price is a key topic for these stakeholders. However, today's online calculators mostly estimate the price of a used car based on age and on the retail price of the vehicle. Other depreciation calculators use straight-line depreciation (the simplest method). Even though this approach might be interesting to calculate the average price of a cohort of similar vehicles, it does not take into account the differences among similar cars. Another approach consists of consulting an expert for his/her appraisal of the vehicle, which is accurate but costly. And last but not least, it is possible for the savvy buyer/seller to check the listings of similar vehicles both online and in physical stores. However, this strategy is very time-consuming. Using predictive models based on available data can help these different stakeholders better estimate their selling and buying prices for used cars in an affordable and quick way.

User Interface:


Welcome to Car Price Predictor

This app predicts the price of a car you want to sell. Try filling the details below:

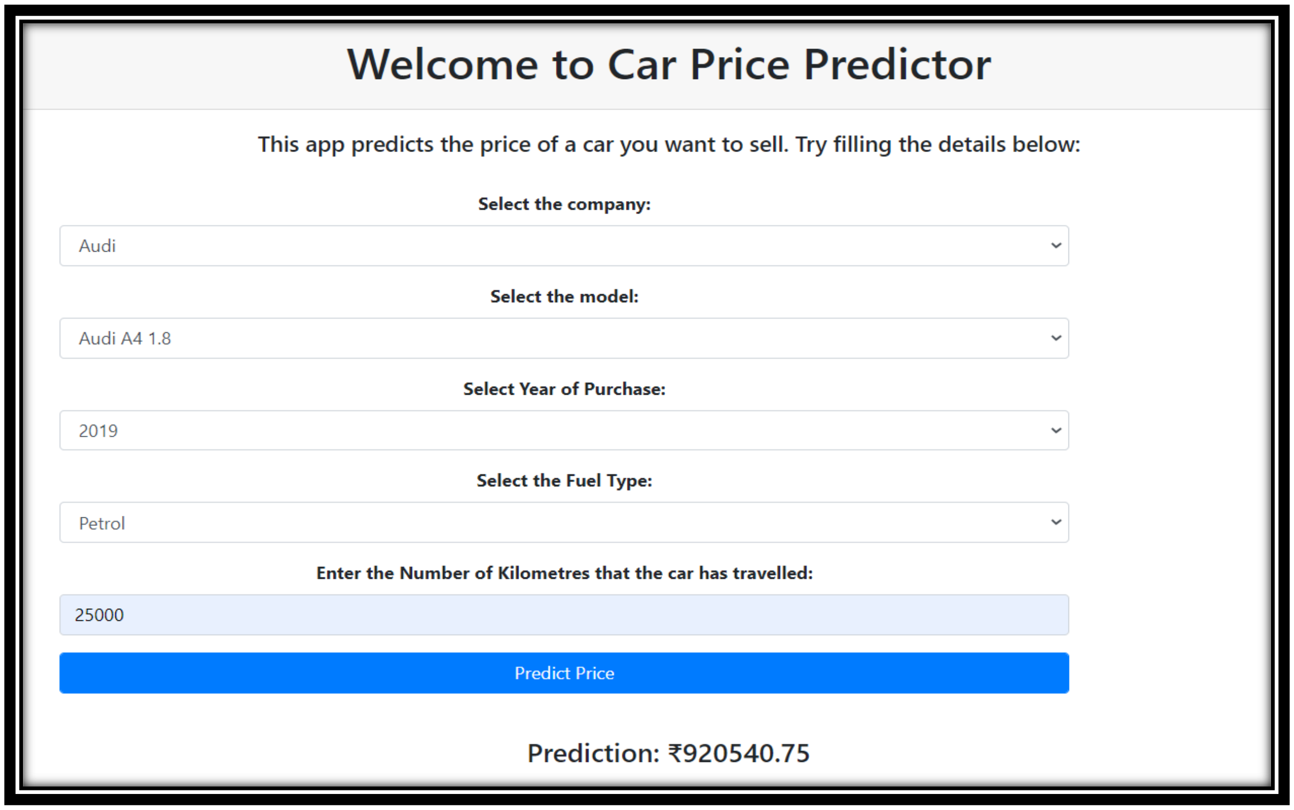
Select the company:
Select Company

Select the model:

Select Year of Purchase:
2019

Select the Fuel Type:
Petrol

Enter the Number of Kilometres that the car has travelled:
Enter the kilometres driven


Welcome to Car Price Predictor

This app predicts the price of a car you want to sell. Try filling the details below:

Select the company:
Audi

Select the model:
Audi A4 1.8

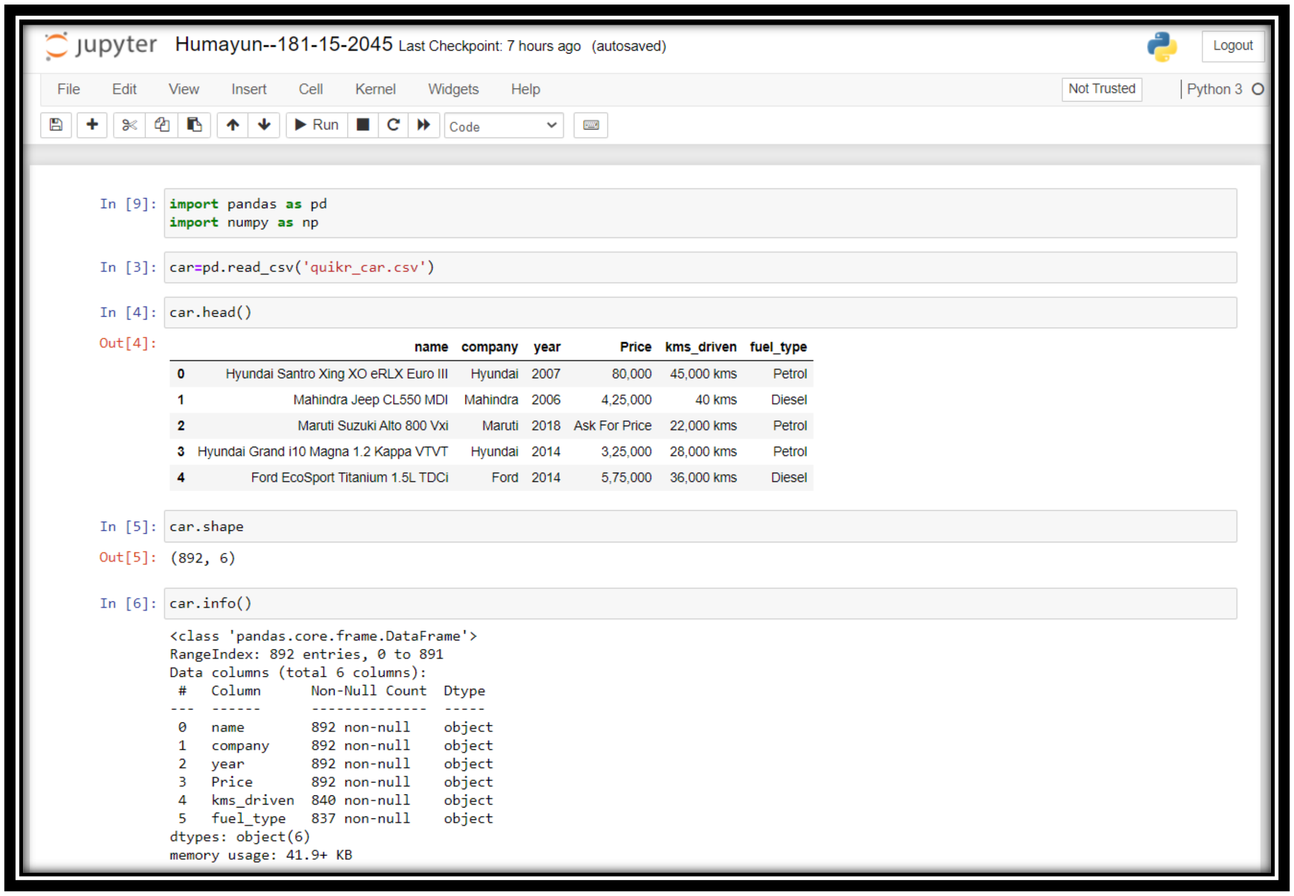
Select Year of Purchase:
2019

Select the Fuel Type:
Petrol

Enter the Number of Kilometres that the car has travelled:
25000

Prediction: ₹920540.75

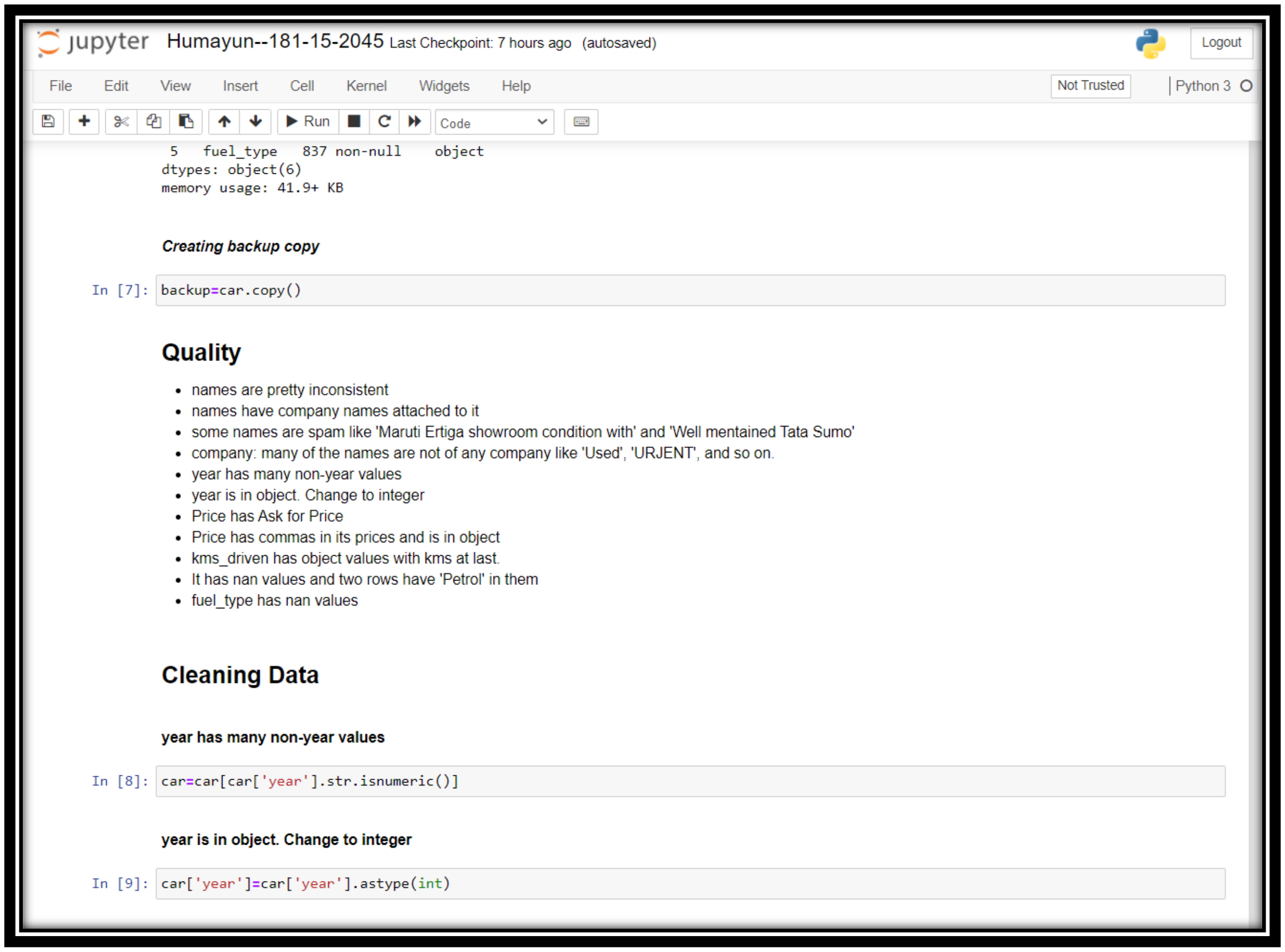
Screenshot of all Console:



```
In [9]: import pandas as pd
        import numpy as np

In [3]: car=pd.read_csv('quikr_car.csv')

In [4]: car.head()
Out[4]:
```

	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing XO eRLX Euro III	Hyundai	2007	80,000	45,000 kms	Petrol
1	Mahindra Jeep CL550 MDI	Mahindra	2006	4,25,000	40 kms	Diesel
2	Maruti Suzuki Alto 800 Vxi	Maruti	2018	Ask For Price	22,000 kms	Petrol
3	Hyundai Grand i10 Magna 1.2 Kappa VTVT	Hyundai	2014	3,25,000	28,000 kms	Petrol
4	Ford EcoSport Titanium 1.5L TDCi	Ford	2014	5,75,000	36,000 kms	Diesel

```
In [5]: car.shape
Out[5]: (892, 6)

In [6]: car.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 892 entries, 0 to 891
        Data columns (total 6 columns):
         #   Column      Non-Null Count  Dtype
        ---  ------      --------------  -----
         0   name        892 non-null    object
         1   company     892 non-null    object
         2   year        892 non-null    object
         3   Price       892 non-null    object
         4   kms_driven  840 non-null    object
         5   fuel_type   837 non-null    object
        dtypes: object(6)
        memory usage: 41.9+ KB
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
 5   fuel_type   837 non-null    object
dtypes: object(6)
memory usage: 41.9+ KB
```

Creating backup copy

```
In [7]: backup=car.copy()
```

Quality

- names are pretty inconsistent
- names have company names attached to it
- some names are spam like 'Maruti Ertiga showroom condition with' and 'Well mentained Tata Sumo'
- company: many of the names are not of any company like 'Used', 'URJENT', and so on.
- year has many non-year values
- year is in object. Change to integer
- Price has Ask for Price
- Price has commas in its prices and is in object
- kms_driven has object values with kms at last.
- It has nan values and two rows have 'Petrol' in them
- fuel_type has nan values

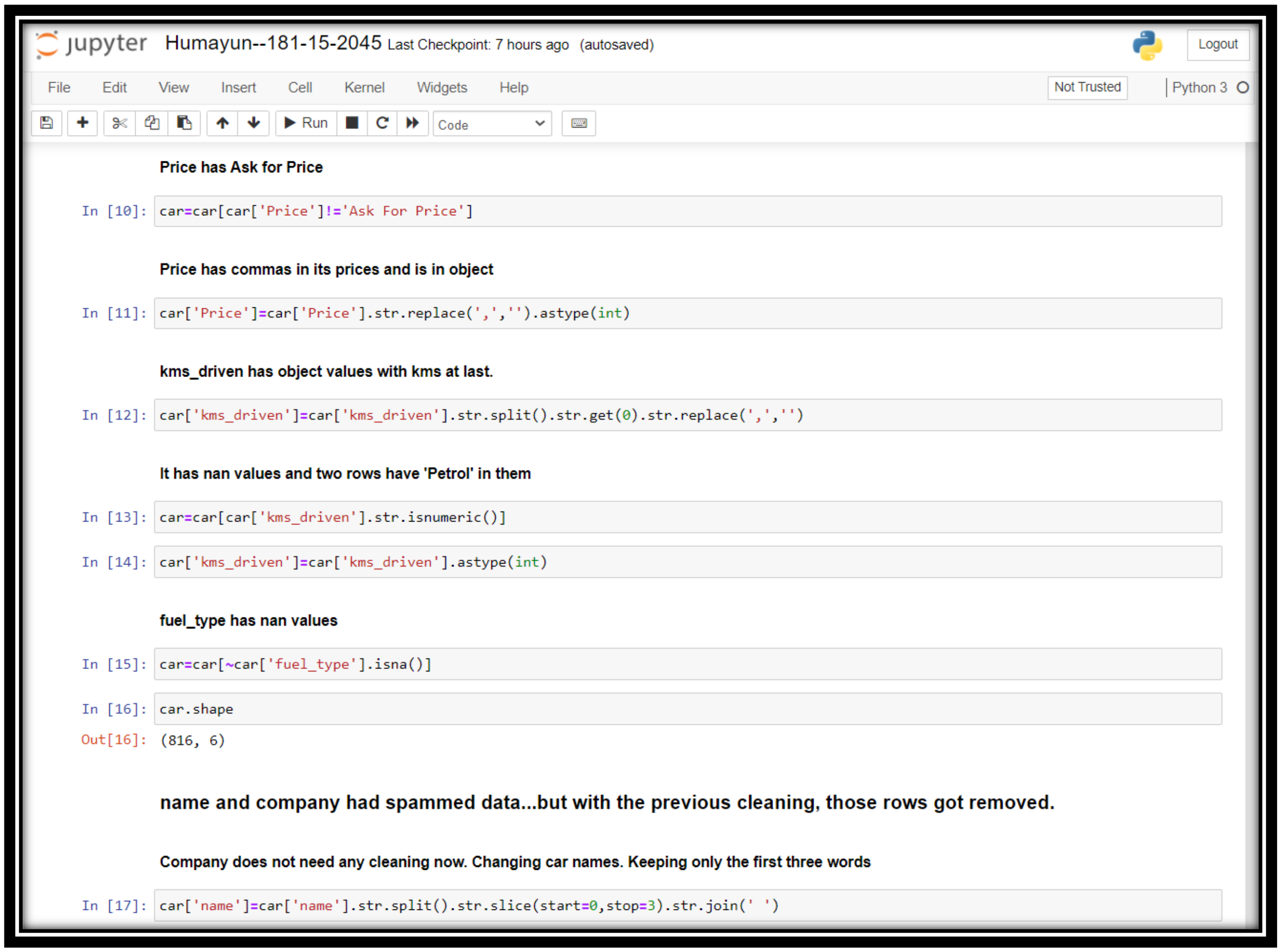
Cleaning Data

year has many non-year values

```
In [8]: car=car[car['year'].str.isnumeric()]
```

year is in object. Change to integer

```
In [9]: car['year']=car['year'].astype(int)
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

Price has Ask for Price

```
In [10]: car=car[car['Price']!='Ask For Price']
```

Price has commas in its prices and is in object

```
In [11]: car['Price']=car['Price'].str.replace(',','').astype(int)
```

kms_driven has object values with kms at last.

```
In [12]: car['kms_driven']=car['kms_driven'].str.split().str.get(0).str.replace(',','')
```

It has nan values and two rows have 'Petrol' in them

```
In [13]: car=car[car['kms_driven'].str.isnumeric()]
```

```
In [14]: car['kms_driven']=car['kms_driven'].astype(int)
```

fuel_type has nan values

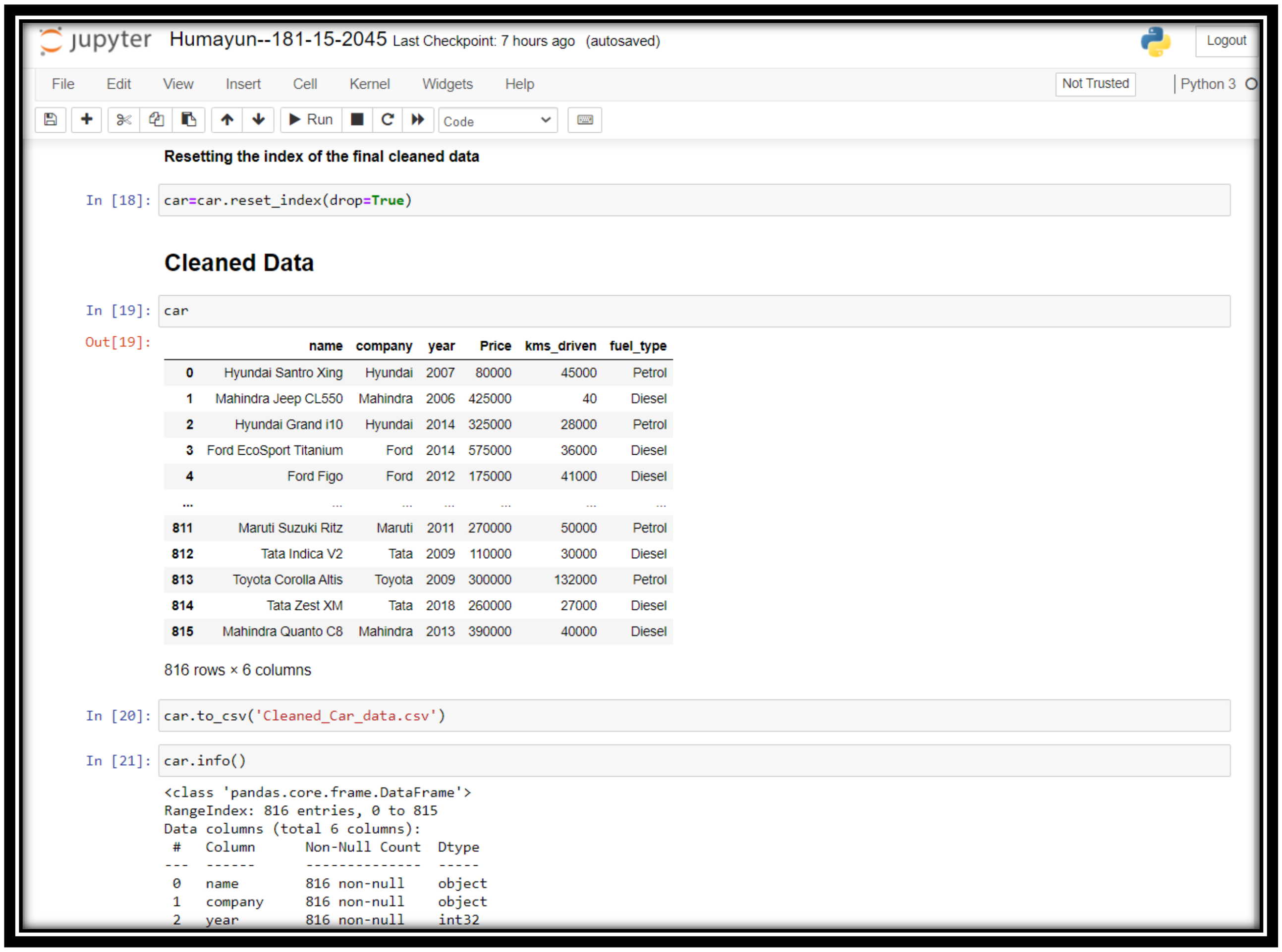
```
In [15]: car=car[~car['fuel_type'].isna()]
```

```
In [16]: car.shape
Out[16]: (816, 6)
```

name and company had spammed data...but with the previous cleaning, those rows got removed.

Company does not need any cleaning now. Changing car names. Keeping only the first three words

```
In [17]: car['name']=car['name'].str.split().str.slice(start=0,stop=3).str.join(' ')
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

Resetting the index of the final cleaned data

```
In [18]: car=car.reset_index(drop=True)
```

Cleaned Data

```
In [19]: car
```

Out[19]:

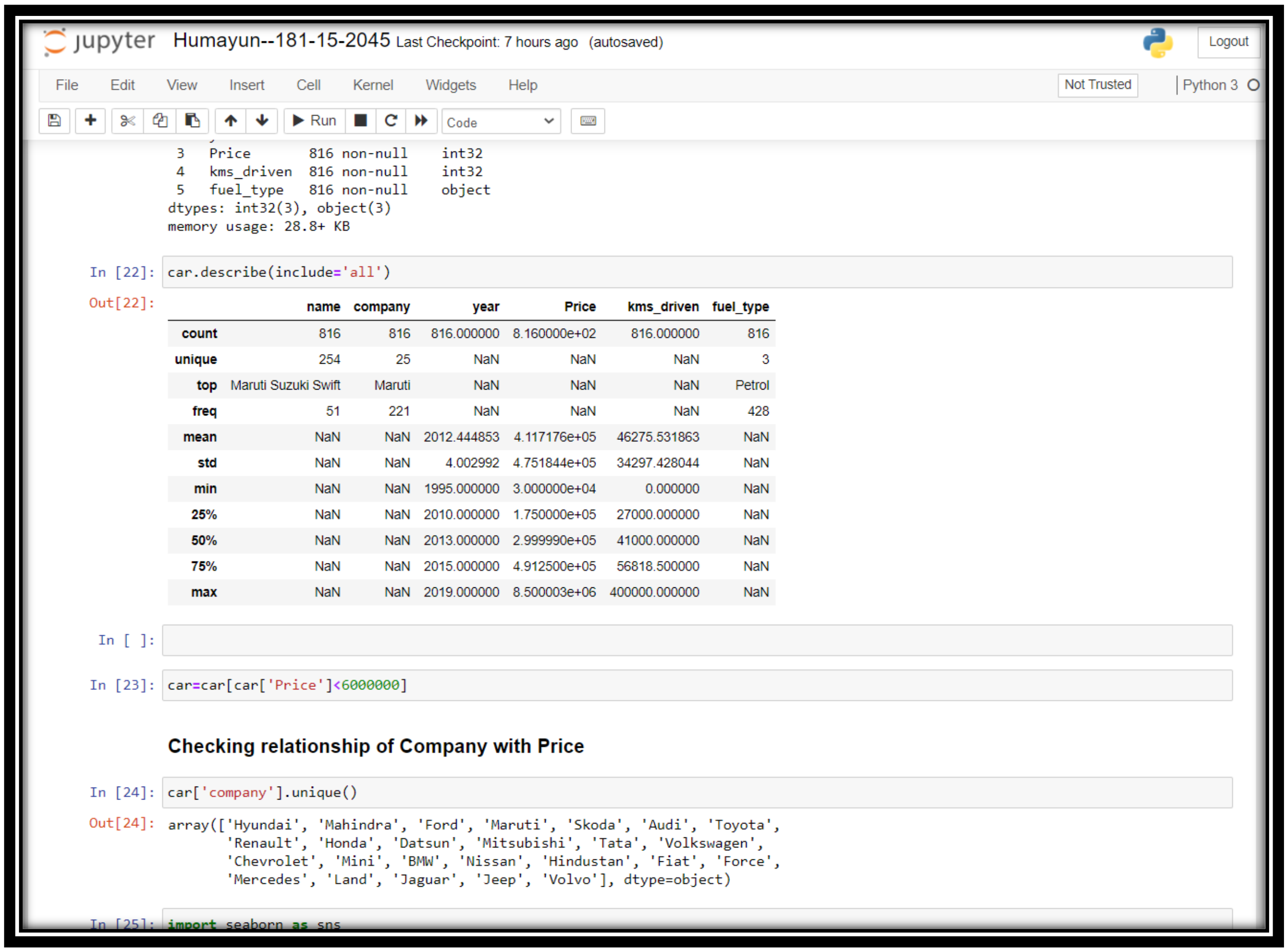
	name	company	year	Price	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	80000	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	425000	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	325000	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	575000	36000	Diesel
4	Ford Figo	Ford	2012	175000	41000	Diesel
...	...	...	...	...	...	...
811	Maruti Suzuki Ritz	Maruti	2011	270000	50000	Petrol
812	Tata Indica V2	Tata	2009	110000	30000	Diesel
813	Toyota Corolla Altis	Toyota	2009	300000	132000	Petrol
814	Tata Zest XM	Tata	2018	260000	27000	Diesel
815	Mahindra Quanto C8	Mahindra	2013	390000	40000	Diesel

816 rows × 6 columns

```
In [20]: car.to_csv('Cleaned_Car_data.csv')
```

```
In [21]: car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         816 non-null    object
 1   company      816 non-null    object
 2   year         816 non-null    int32
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
 3   Price      816 non-null   int32
 4   kms_driven 816 non-null   int32
 5   fuel_type  816 non-null   object
dtypes: int32(3), object(3)
memory usage: 28.8+ KB
```

In [22]:
```
car.describe(include='all')
```

Out[22]:

	name	company	year	Price	kms_driven	fuel_type
count	816	816	816.000000	8.160000e+02	816.000000	816
unique	254	25	NaN	NaN	NaN	3
top	Maruti Suzuki Swift	Maruti	NaN	NaN	NaN	Petrol
freq	51	221	NaN	NaN	NaN	428
mean	NaN	NaN	2012.444853	4.117176e+05	46275.531863	NaN
std	NaN	NaN	4.002992	4.751844e+05	34297.428044	NaN
min	NaN	NaN	1995.000000	3.000000e+04	0.000000	NaN
25%	NaN	NaN	2010.000000	1.750000e+05	27000.000000	NaN
50%	NaN	NaN	2013.000000	2.999990e+05	41000.000000	NaN
75%	NaN	NaN	2015.000000	4.912500e+05	56818.500000	NaN
max	NaN	NaN	2019.000000	8.500003e+06	400000.000000	NaN

In [ ]:

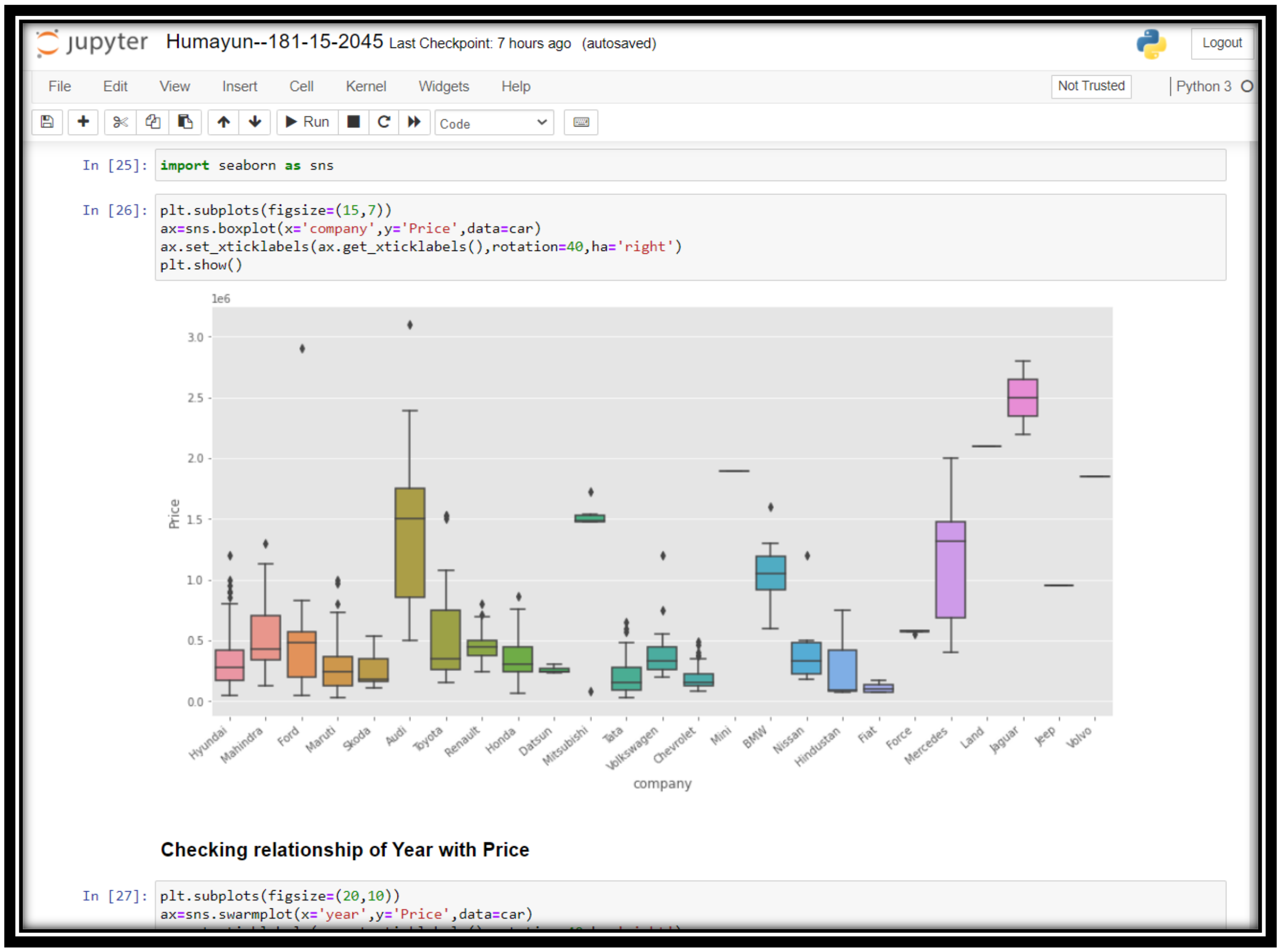
In [23]:
```
car=car[car['Price']<6000000]
```

Checking relationship of Company with Price

In [24]:
```
car['company'].unique()
```

Out[24]:
```
array(['Hyundai', 'Mahindra', 'Ford', 'Maruti', 'Skoda', 'Audi', 'Toyota',
       'Renault', 'Honda', 'Datsun', 'Mitsubishi', 'Tata', 'Volkswagen',
       'Chevrolet', 'Mini', 'BMW', 'Nissan', 'Hindustan', 'Fiat', 'Force',
       'Mercedes', 'Land', 'Jaguar', 'Jeep', 'Volvo'], dtype=object)
```

In [25]:
```
import seaborn as sns
```

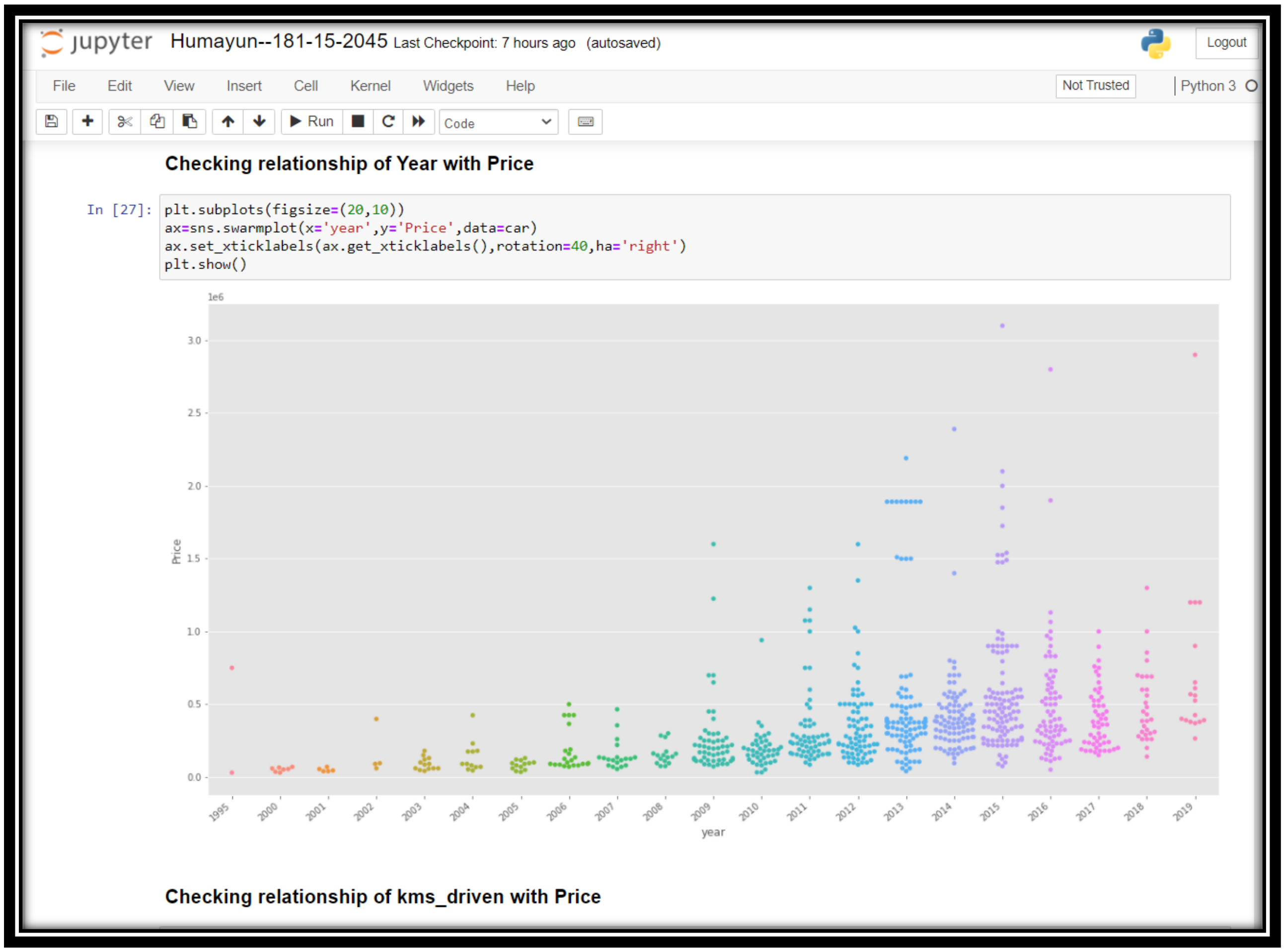


In [25]:
```
import seaborn as sns
```

In [26]:
```
plt.subplots(figsize=(15,7))
ax=sns.boxplot(x='company',y='Price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```

Checking relationship of Year with Price

In [27]:
```
plt.subplots(figsize=(20,10))
ax=sns.swarmplot(x='year',y='Price',data=car)
```

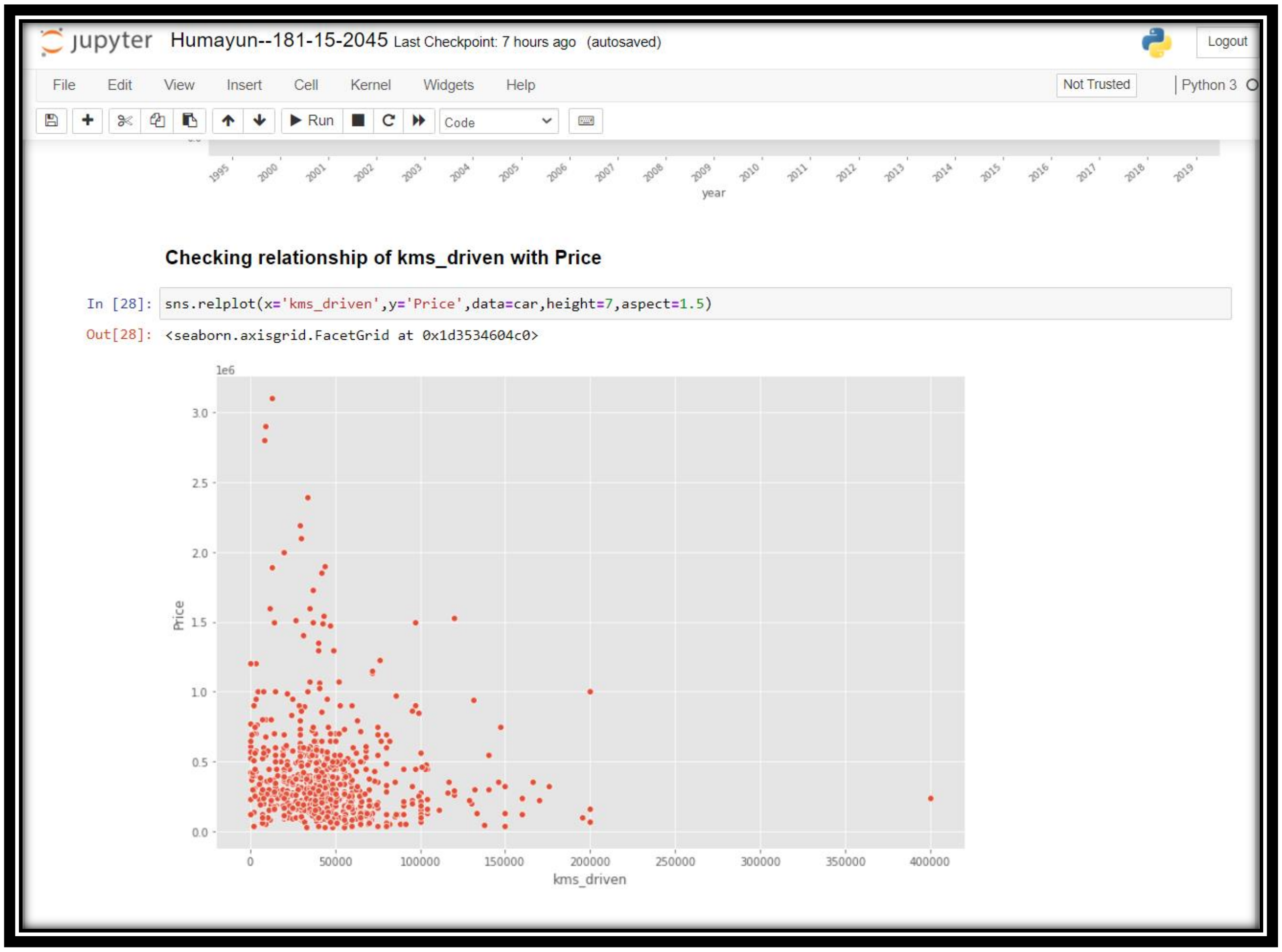


jupyter
Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

Checking relationship of Year with Price

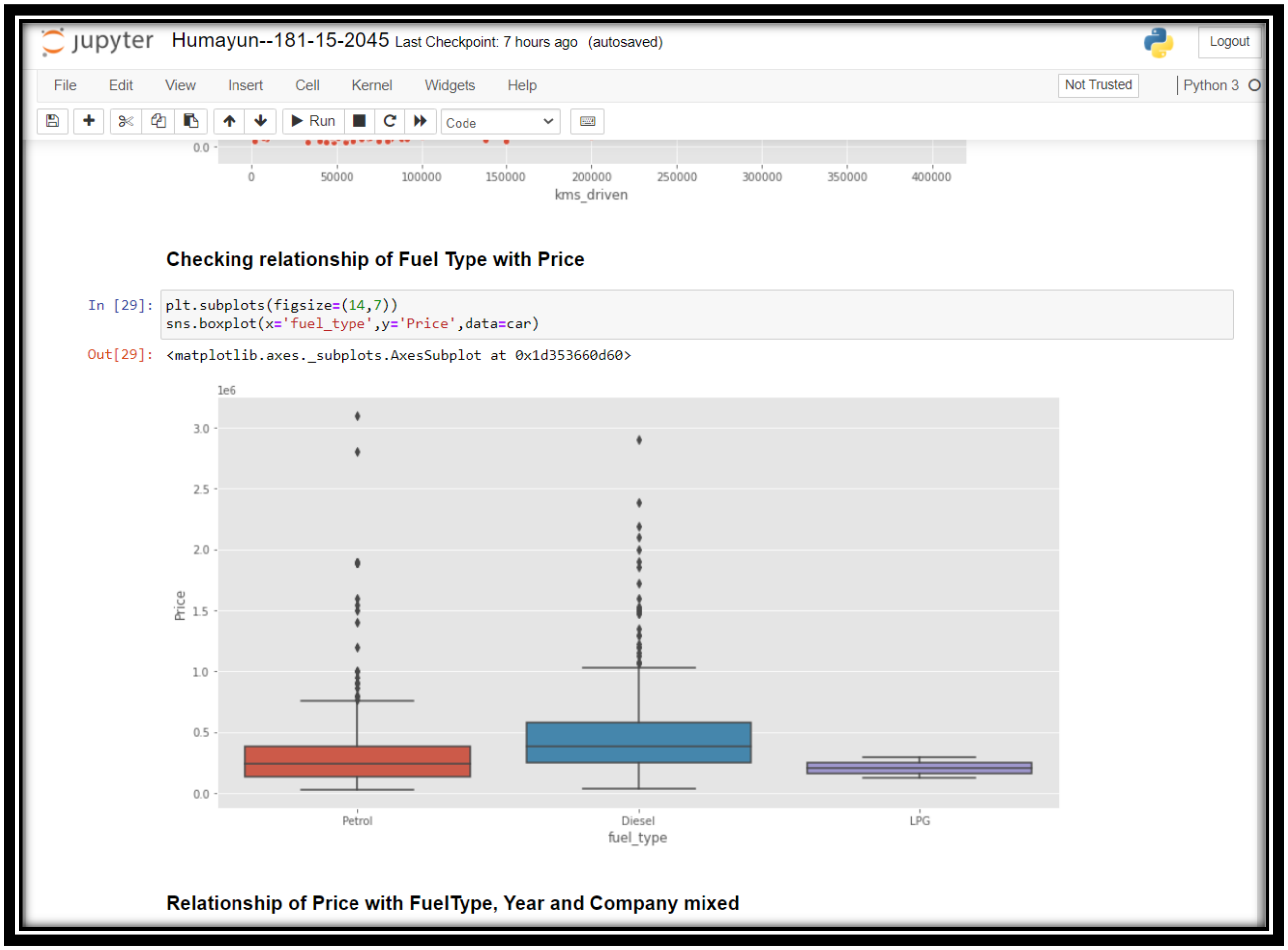
In [27]:
```
plt.subplots(figsize=(20,10))
ax=sns.swarmplot(x='year',y='Price',data=car)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```

Checking relationship of kms_driven with Price


jupyter
Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
year

Checking relationship of kms_driven with Price

```
In [28]: sns.relplot(x='kms_driven',y='Price',data=car,height=7,aspect=1.5)
Out[28]: <seaborn.axisgrid.FacetGrid at 0x1d3534604c0>
```

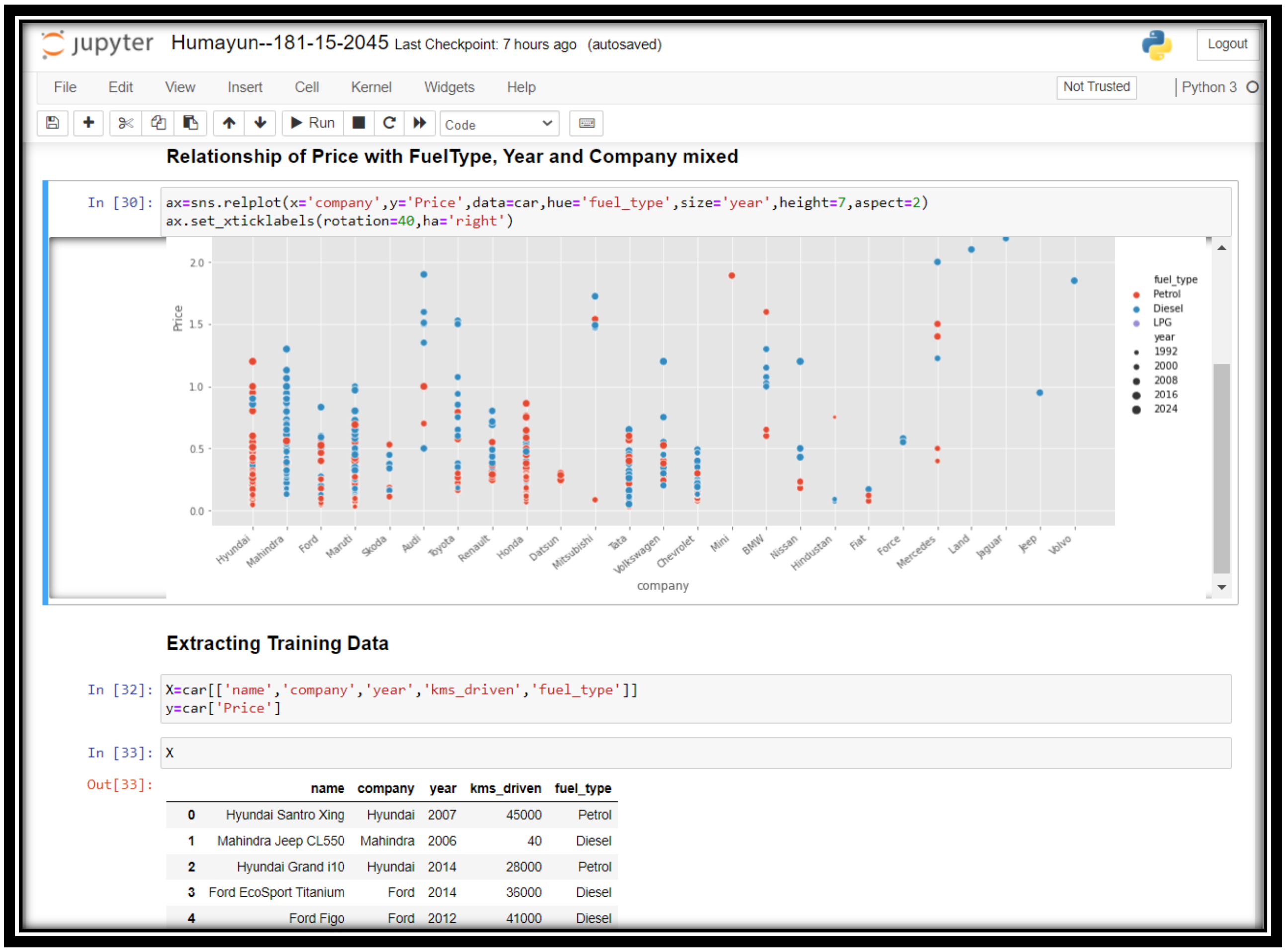


Checking relationship of Fuel Type with Price

```
In [29]: plt.subplots(figsize=(14,7))
         sns.boxplot(x='fuel_type',y='Price',data=car)
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1d353660d60>
```

Relationship of Price with FuelType, Year and Company mixed



Relationship of Price with FuelType, Year and Company mixed

```
In [30]: ax=sns.relplot(x='company',y='Price',data=car,hue='fuel_type',size='year',height=7,aspect=2)
         ax.set_xticklabels(rotation=40,ha='right')
```

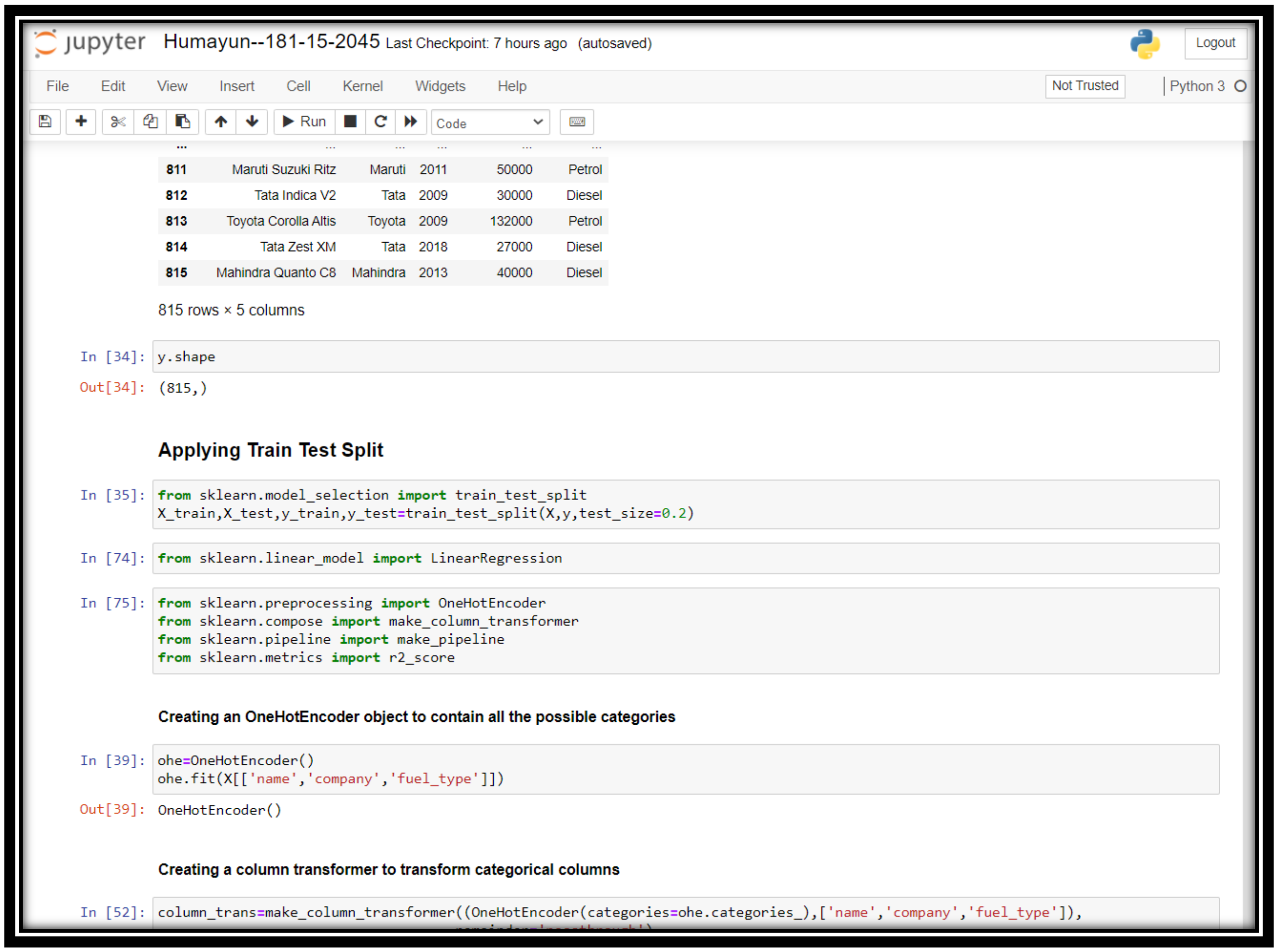
Extracting Training Data

```
In [32]: X=car[['name','company','year','kms_driven','fuel_type']]
         y=car['Price']
```

```
In [33]: X
```

Out[33]:

	name	company	year	kms_driven	fuel_type
0	Hyundai Santro Xing	Hyundai	2007	45000	Petrol
1	Mahindra Jeep CL550	Mahindra	2006	40	Diesel
2	Hyundai Grand i10	Hyundai	2014	28000	Petrol
3	Ford EcoSport Titanium	Ford	2014	36000	Diesel
4	Ford Figo	Ford	2012	41000	Diesel



jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)

...	...	...	...	...	...
811	Maruti Suzuki Ritz	Maruti	2011	50000	Petrol
812	Tata Indica V2	Tata	2009	30000	Diesel
813	Toyota Corolla Altis	Toyota	2009	132000	Petrol
814	Tata Zest XM	Tata	2018	27000	Diesel
815	Mahindra Quanto C8	Mahindra	2013	40000	Diesel

815 rows × 5 columns

```
In [34]: y.shape
Out[34]: (815,)
```

Applying Train Test Split

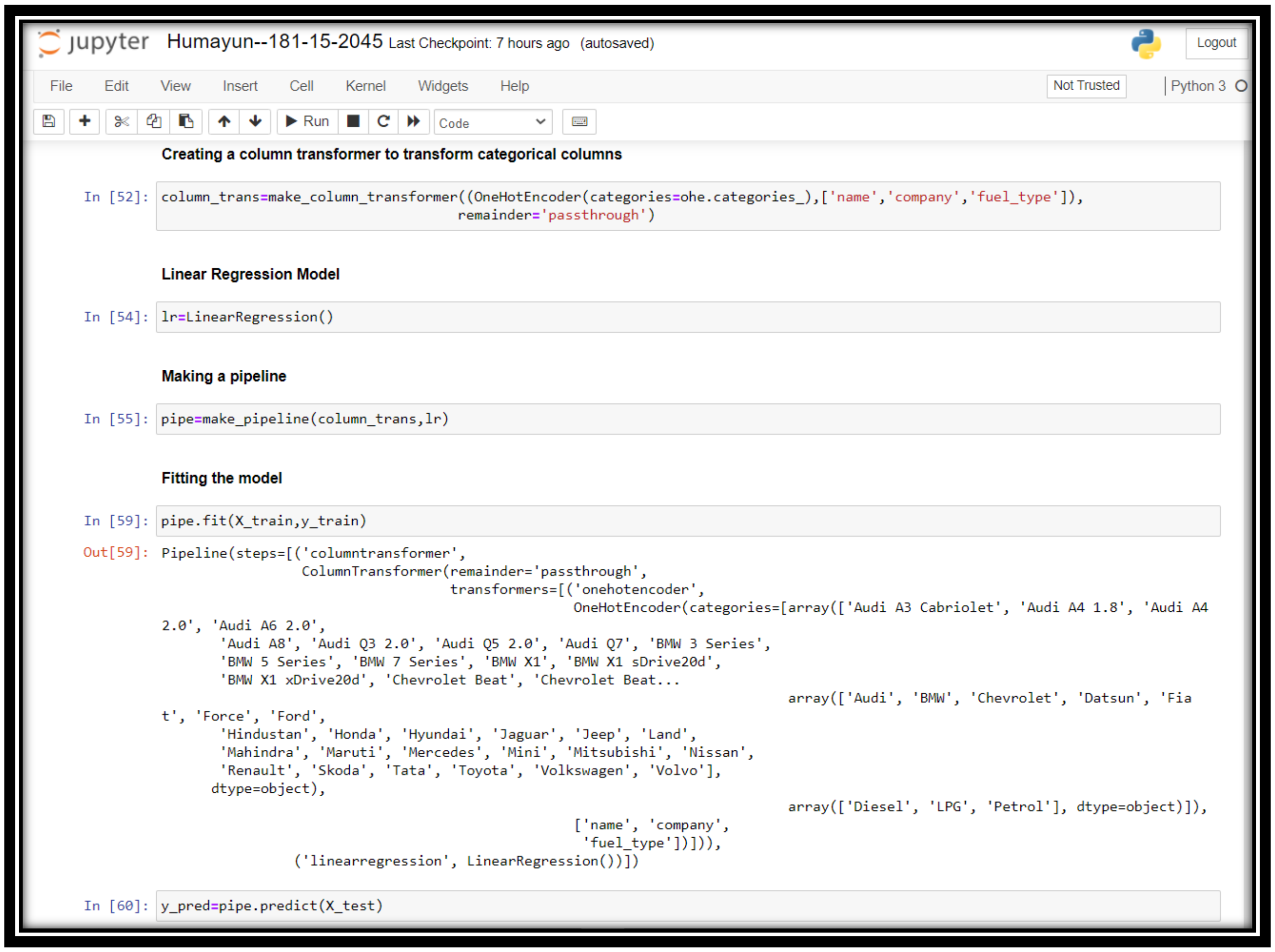
```
In [35]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

In [74]: from sklearn.linear_model import LinearRegression

In [75]: from sklearn.preprocessing import OneHotEncoder
         from sklearn.compose import make_column_transformer
         from sklearn.pipeline import make_pipeline
         from sklearn.metrics import r2_score
```

Creating an OneHotEncoder object to contain all the possible categories

```
In [39]: ohe=OneHotEncoder()
         ohe.fit(X[['name','company','fuel_type']])
Out[39]: OneHotEncoder()
```

Creating a column transformer to transform categorical columns

```
In [52]: column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)

Creating a column transformer to transform categorical columns

```
In [52]: column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                    remainder='passthrough')
```

Linear Regression Model

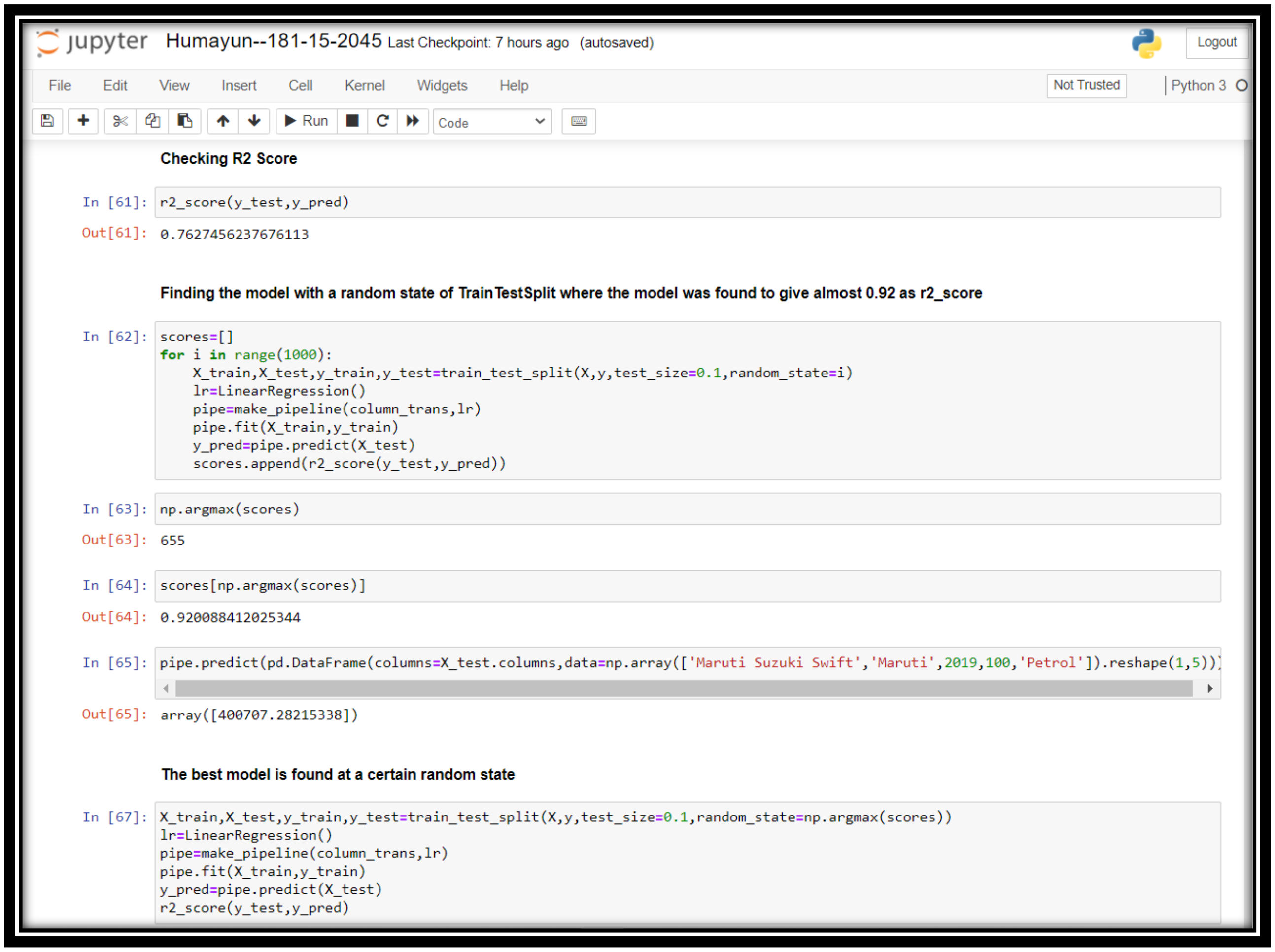
```
In [54]: lr=LinearRegression()
```

Making a pipeline

```
In [55]: pipe=make_pipeline(column_trans,lr)
```

Fitting the model

```
In [59]: pipe.fit(X_train,y_train)

Out[59]: Pipeline(steps=[('columntransformer',
                          ColumnTransformer(remainder='passthrough',
                                            transformers=[('onehotencoder',
                                                           OneHotEncoder(categories=[array(['Audi A3 Cabriolet', 'Audi A4 1.8', 'Audi A4
         2.0', 'Audi A6 2.0',
                'Audi A8', 'Audi Q3 2.0', 'Audi Q5 2.0', 'Audi Q7', 'BMW 3 Series',
                'BMW 5 Series', 'BMW 7 Series', 'BMW X1', 'BMW X1 sDrive20d',
                'BMW X1 xDrive20d', 'Chevrolet Beat', 'Chevrolet Beat...
                                                                                     array(['Audi', 'BMW', 'Chevrolet', 'Datsun', 'Fia
         t', 'Force', 'Ford',
                'Hindustan', 'Honda', 'Hyundai', 'Jaguar', 'Jeep', 'Land',
                'Mahindra', 'Maruti', 'Mercedes', 'Mini', 'Mitsubishi', 'Nissan',
                'Renault', 'Skoda', 'Tata', 'Toyota', 'Volkswagen', 'Volvo'],
               dtype=object),
                                                                                     array(['Diesel', 'LPG', 'Petrol'], dtype=object)]),
                                                           ['name', 'company',
                                                            'fuel_type'])])),
                         ('linearregression', LinearRegression())])

In [60]: y_pred=pipe.predict(X_test)
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

Checking R2 Score

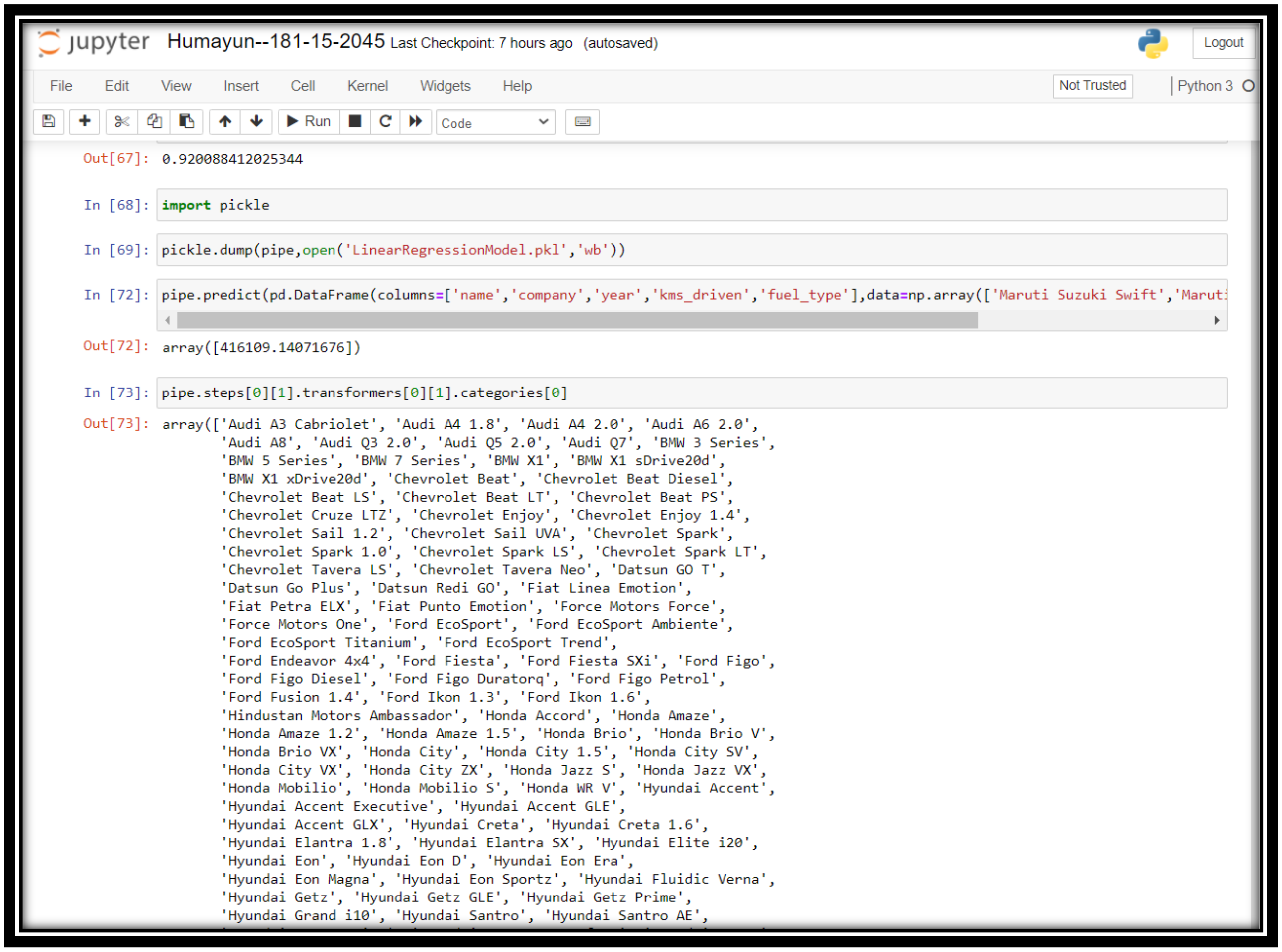
```
In [61]: r2_score(y_test,y_pred)
Out[61]: 0.7627456237676113
```

Finding the model with a random state of TrainTestSplit where the model was found to give almost 0.92 as r2_score

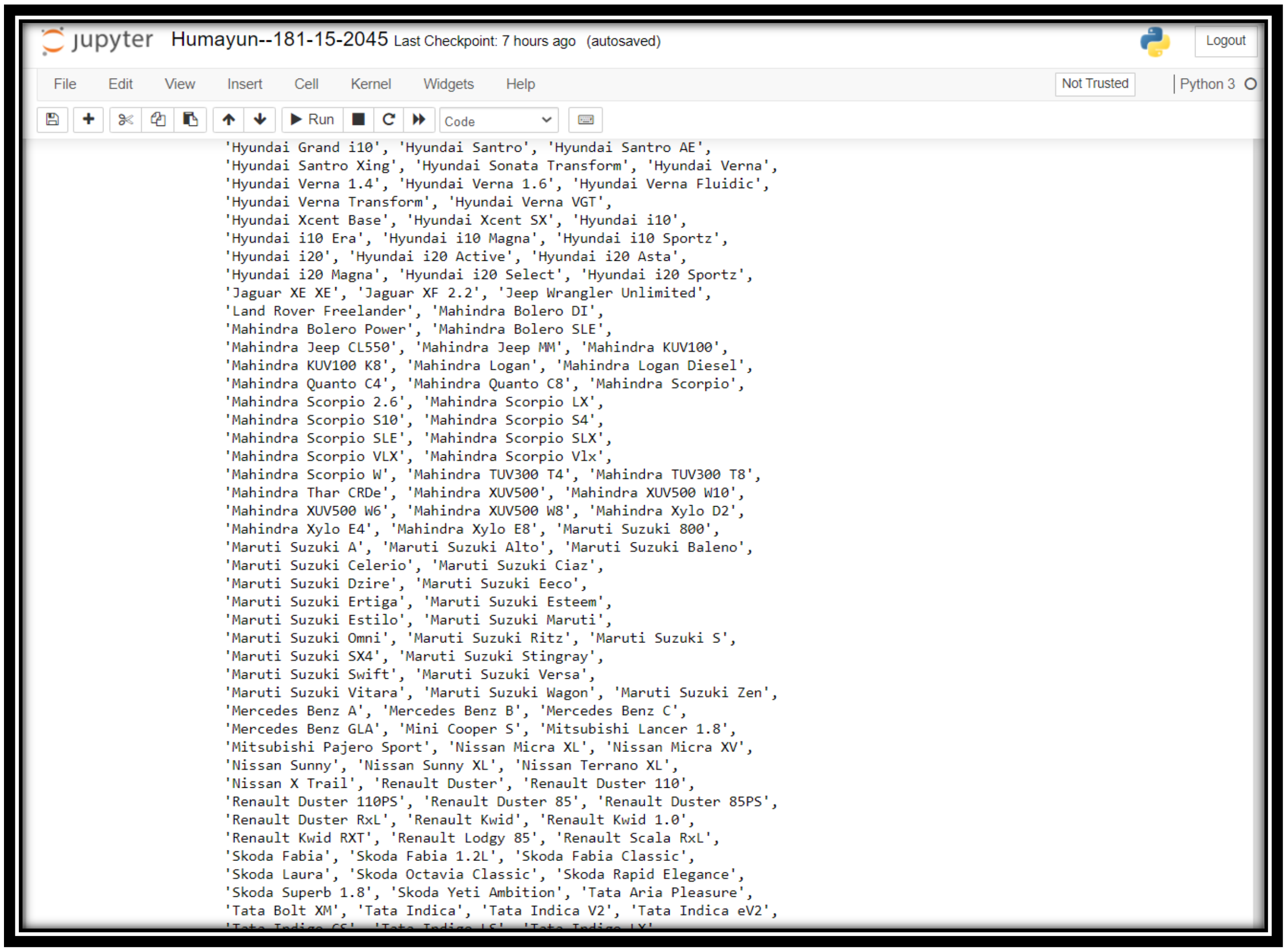
```
In [62]: scores=[]
         for i in range(1000):
             X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
             lr=LinearRegression()
             pipe=make_pipeline(column_trans,lr)
             pipe.fit(X_train,y_train)
             y_pred=pipe.predict(X_test)
             scores.append(r2_score(y_test,y_pred))

In [63]: np.argmax(scores)
Out[63]: 655

In [64]: scores[np.argmax(scores)]
Out[64]: 0.920088412025344

In [65]: pipe.predict(pd.DataFrame(columns=X_test.columns,data=np.array(['Maruti Suzuki Swift','Maruti',2019,100,'Petrol']).reshape(1,5)))
Out[65]: array([400707.28215338])
```

The best model is found at a certain random state

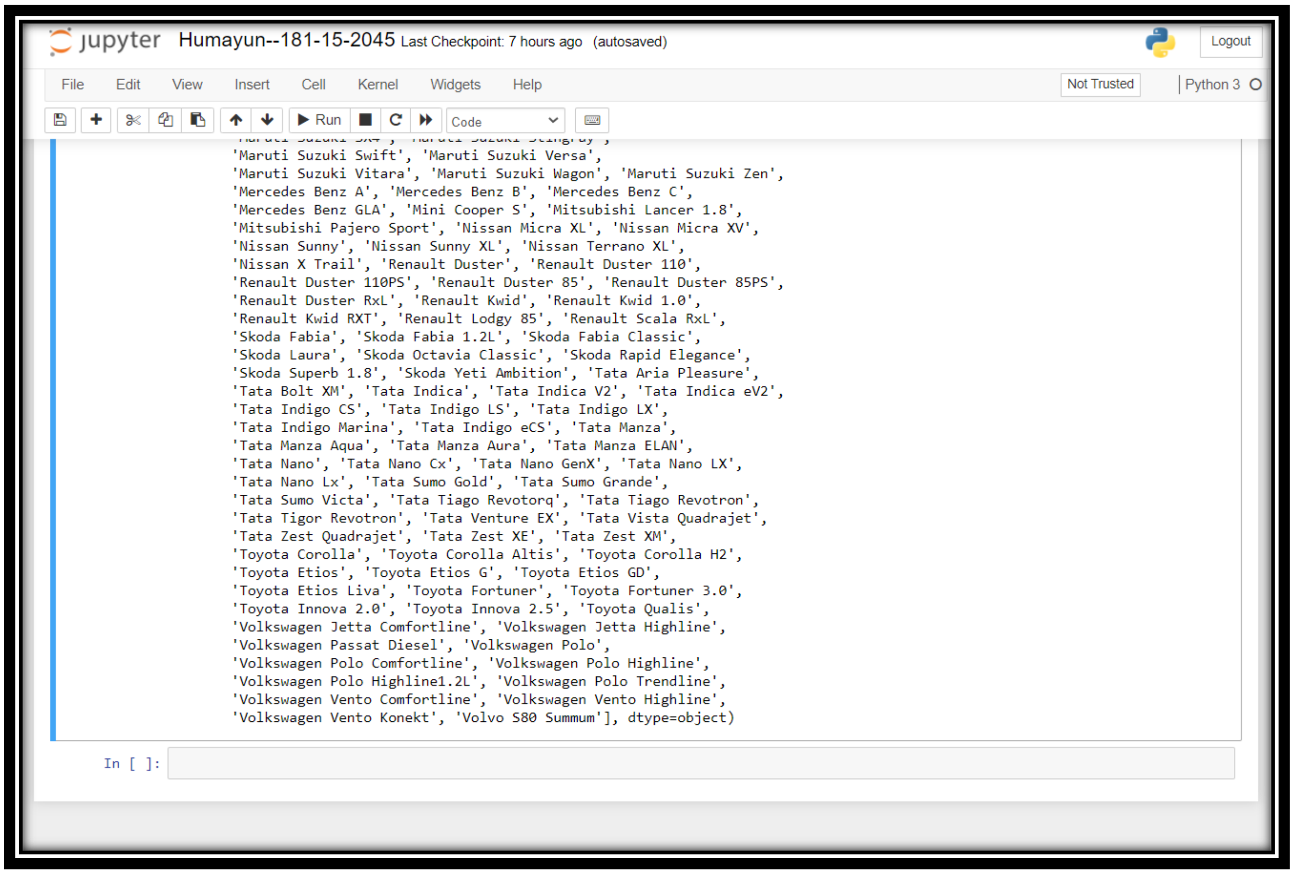
```
In [67]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
         lr=LinearRegression()
         pipe=make_pipeline(column_trans,lr)
         pipe.fit(X_train,y_train)
         y_pred=pipe.predict(X_test)
         r2_score(y_test,y_pred)
```


jupyter Humayun--181-15-2045 Last Checkpoint: 7 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
Out[67]: 0.920088412025344

In [68]: import pickle

In [69]: pickle.dump(pipe,open('LinearRegressionModel.pkl','wb'))

In [72]: pipe.predict(pd.DataFrame(columns=['name','company','year','kms_driven','fuel_type'],data=np.array(['Maruti Suzuki Swift','Marut

Out[72]: array([416109.14071676])

In [73]: pipe.steps[0][1].transformers[0][1].categories[0]

Out[73]: array(['Audi A3 Cabriolet', 'Audi A4 1.8', 'Audi A4 2.0', 'Audi A6 2.0',
                'Audi A8', 'Audi Q3 2.0', 'Audi Q5 2.0', 'Audi Q7', 'BMW 3 Series',
                'BMW 5 Series', 'BMW 7 Series', 'BMW X1', 'BMW X1 sDrive20d',
                'BMW X1 xDrive20d', 'Chevrolet Beat', 'Chevrolet Beat Diesel',
                'Chevrolet Beat LS', 'Chevrolet Beat LT', 'Chevrolet Beat PS',
                'Chevrolet Cruze LTZ', 'Chevrolet Enjoy', 'Chevrolet Enjoy 1.4',
                'Chevrolet Sail 1.2', 'Chevrolet Sail UVA', 'Chevrolet Spark',
                'Chevrolet Spark 1.0', 'Chevrolet Spark LS', 'Chevrolet Spark LT',
                'Chevrolet Tavera LS', 'Chevrolet Tavera Neo', 'Datsun GO T',
                'Datsun Go Plus', 'Datsun Redi GO', 'Fiat Linea Emotion',
                'Fiat Petra ELX', 'Fiat Punto Emotion', 'Force Motors Force',
                'Force Motors One', 'Ford EcoSport', 'Ford EcoSport Ambiente',
                'Ford EcoSport Titanium', 'Ford EcoSport Trend',
                'Ford Endeavor 4x4', 'Ford Fiesta', 'Ford Fiesta SXi', 'Ford Figo',
                'Ford Figo Diesel', 'Ford Figo Duratorq', 'Ford Figo Petrol',
                'Ford Fusion 1.4', 'Ford Ikon 1.3', 'Ford Ikon 1.6',
                'Hindustan Motors Ambassador', 'Honda Accord', 'Honda Amaze',
                'Honda Amaze 1.2', 'Honda Amaze 1.5', 'Honda Brio', 'Honda Brio V',
                'Honda Brio VX', 'Honda City', 'Honda City 1.5', 'Honda City SV',
                'Honda City VX', 'Honda City ZX', 'Honda Jazz S', 'Honda Jazz VX',
                'Honda Mobilio', 'Honda Mobilio S', 'Honda WR V', 'Hyundai Accent',
                'Hyundai Accent Executive', 'Hyundai Accent GLE',
                'Hyundai Accent GLX', 'Hyundai Creta', 'Hyundai Creta 1.6',
                'Hyundai Elantra 1.8', 'Hyundai Elantra SX', 'Hyundai Elite i20',
                'Hyundai Eon', 'Hyundai Eon D', 'Hyundai Eon Era',
                'Hyundai Eon Magna', 'Hyundai Eon Sportz', 'Hyundai Fluidic Verna',
                'Hyundai Getz', 'Hyundai Getz GLE', 'Hyundai Getz Prime',
                'Hyundai Grand i10', 'Hyundai Santro', 'Hyundai Santro AE',
```



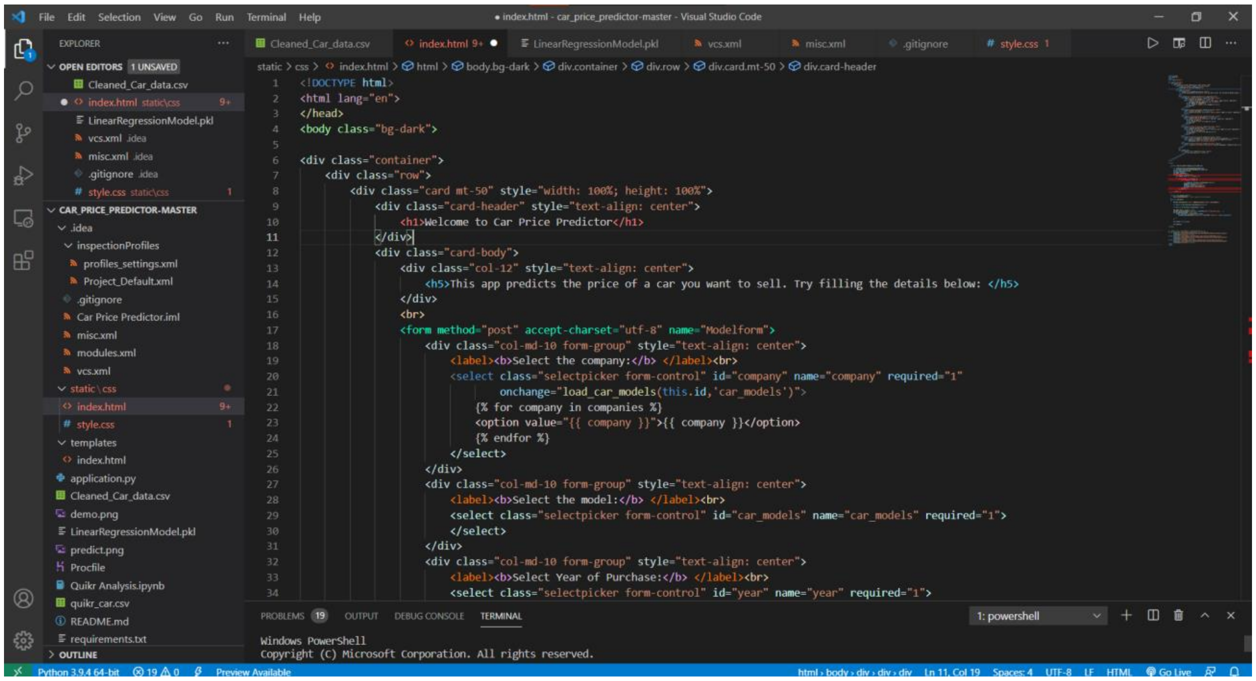
```
'Hyundai Grand i10', 'Hyundai Santro', 'Hyundai Santro AE',
'Hyundai Santro Xing', 'Hyundai Sonata Transform', 'Hyundai Verna',
'Hyundai Verna 1.4', 'Hyundai Verna 1.6', 'Hyundai Verna Fluidic',
'Hyundai Verna Transform', 'Hyundai Verna VGT',
'Hyundai Xcent Base', 'Hyundai Xcent SX', 'Hyundai i10',
'Hyundai i10 Era', 'Hyundai i10 Magna', 'Hyundai i10 Sportz',
'Hyundai i20', 'Hyundai i20 Active', 'Hyundai i20 Asta',
'Hyundai i20 Magna', 'Hyundai i20 Select', 'Hyundai i20 Sportz',
'Jaguar XE XE', 'Jaguar XF 2.2', 'Jeep Wrangler Unlimited',
'Land Rover Freelander', 'Mahindra Bolero DI',
'Mahindra Bolero Power', 'Mahindra Bolero SLE',
'Mahindra Jeep CL550', 'Mahindra Jeep MM', 'Mahindra KUV100',
'Mahindra KUV100 K8', 'Mahindra Logan', 'Mahindra Logan Diesel',
'Mahindra Quanto C4', 'Mahindra Quanto C8', 'Mahindra Scorpio',
'Mahindra Scorpio 2.6', 'Mahindra Scorpio LX',
'Mahindra Scorpio S10', 'Mahindra Scorpio S4',
'Mahindra Scorpio SLE', 'Mahindra Scorpio SLX',
'Mahindra Scorpio VLX', 'Mahindra Scorpio Vlx',
'Mahindra Scorpio W', 'Mahindra TUV300 T4', 'Mahindra TUV300 T8',
'Mahindra Thar CRDe', 'Mahindra XUV500', 'Mahindra XUV500 W10',
'Mahindra XUV500 W6', 'Mahindra XUV500 W8', 'Mahindra Xylo D2',
'Mahindra Xylo E4', 'Mahindra Xylo E8', 'Maruti Suzuki 800',
'Maruti Suzuki A', 'Maruti Suzuki Alto', 'Maruti Suzuki Baleno',
'Maruti Suzuki Celerio', 'Maruti Suzuki Ciaz',
'Maruti Suzuki Dzire', 'Maruti Suzuki Eeco',
'Maruti Suzuki Ertiga', 'Maruti Suzuki Esteem',
'Maruti Suzuki Estilo', 'Maruti Suzuki Maruti',
'Maruti Suzuki Omni', 'Maruti Suzuki Ritz', 'Maruti Suzuki S',
'Maruti Suzuki SX4', 'Maruti Suzuki Stingray',
'Maruti Suzuki Swift', 'Maruti Suzuki Versa',
'Maruti Suzuki Vitara', 'Maruti Suzuki Wagon', 'Maruti Suzuki Zen',
'Mercedes Benz A', 'Mercedes Benz B', 'Mercedes Benz C',
'Mercedes Benz GLA', 'Mini Cooper S', 'Mitsubishi Lancer 1.8',
'Mitsubishi Pajero Sport', 'Nissan Micra XL', 'Nissan Micra XV',
'Nissan Sunny', 'Nissan Sunny XL', 'Nissan Terrano XL',
'Nissan X Trail', 'Renault Duster', 'Renault Duster 110',
'Renault Duster 110PS', 'Renault Duster 85', 'Renault Duster 85PS',
'Renault Duster RxL', 'Renault Kwid', 'Renault Kwid 1.0',
'Renault Kwid RXT', 'Renault Lodgy 85', 'Renault Scala RxL',
'Skoda Fabia', 'Skoda Fabia 1.2L', 'Skoda Fabia Classic',
'Skoda Laura', 'Skoda Octavia Classic', 'Skoda Rapid Elegance',
'Skoda Superb 1.8', 'Skoda Yeti Ambition', 'Tata Aria Pleasure',
'Tata Bolt XM', 'Tata Indica', 'Tata Indica V2', 'Tata Indica eV2',
'Tata Indigo CS', 'Tata Indigo LS', 'Tata Indigo LX',
```



```
       'Maruti Suzuki Swift', 'Maruti Suzuki Versa',
       'Maruti Suzuki Vitara', 'Maruti Suzuki Wagon', 'Maruti Suzuki Zen',
       'Mercedes Benz A', 'Mercedes Benz B', 'Mercedes Benz C',
       'Mercedes Benz GLA', 'Mini Cooper S', 'Mitsubishi Lancer 1.8',
       'Mitsubishi Pajero Sport', 'Nissan Micra XL', 'Nissan Micra XV',
       'Nissan Sunny', 'Nissan Sunny XL', 'Nissan Terrano XL',
       'Nissan X Trail', 'Renault Duster', 'Renault Duster 110',
       'Renault Duster 110PS', 'Renault Duster 85', 'Renault Duster 85PS',
       'Renault Duster RxL', 'Renault Kwid', 'Renault Kwid 1.0',
       'Renault Kwid RXT', 'Renault Lodgy 85', 'Renault Scala RxL',
       'Skoda Fabia', 'Skoda Fabia 1.2L', 'Skoda Fabia Classic',
       'Skoda Laura', 'Skoda Octavia Classic', 'Skoda Rapid Elegance',
       'Skoda Superb 1.8', 'Skoda Yeti Ambition', 'Tata Aria Pleasure',
       'Tata Bolt XM', 'Tata Indica', 'Tata Indica V2', 'Tata Indica eV2',
       'Tata Indigo CS', 'Tata Indigo LS', 'Tata Indigo LX',
       'Tata Indigo Marina', 'Tata Indigo eCS', 'Tata Manza',
       'Tata Manza Aqua', 'Tata Manza Aura', 'Tata Manza ELAN',
       'Tata Nano', 'Tata Nano Cx', 'Tata Nano GenX', 'Tata Nano LX',
       'Tata Nano Lx', 'Tata Sumo Gold', 'Tata Sumo Grande',
       'Tata Sumo Victa', 'Tata Tiago Revotorq', 'Tata Tiago Revotron',
       'Tata Tigor Revotron', 'Tata Venture EX', 'Tata Vista Quadrajet',
       'Tata Zest Quadrajet', 'Tata Zest XE', 'Tata Zest XM',
       'Toyota Corolla', 'Toyota Corolla Altis', 'Toyota Corolla H2',
       'Toyota Etios', 'Toyota Etios G', 'Toyota Etios GD',
       'Toyota Etios Liva', 'Toyota Fortuner', 'Toyota Fortuner 3.0',
       'Toyota Innova 2.0', 'Toyota Innova 2.5', 'Toyota Qualis',
       'Volkswagen Jetta Comfortline', 'Volkswagen Jetta Highline',
       'Volkswagen Passat Diesel', 'Volkswagen Polo',
       'Volkswagen Polo Comfortline', 'Volkswagen Polo Highline',
       'Volkswagen Polo Highline1.2L', 'Volkswagen Polo Trendline',
       'Volkswagen Vento Comfortline', 'Volkswagen Vento Highline',
       'Volkswagen Vento Konekt', 'Volvo S80 Summum'], dtype=object)
```

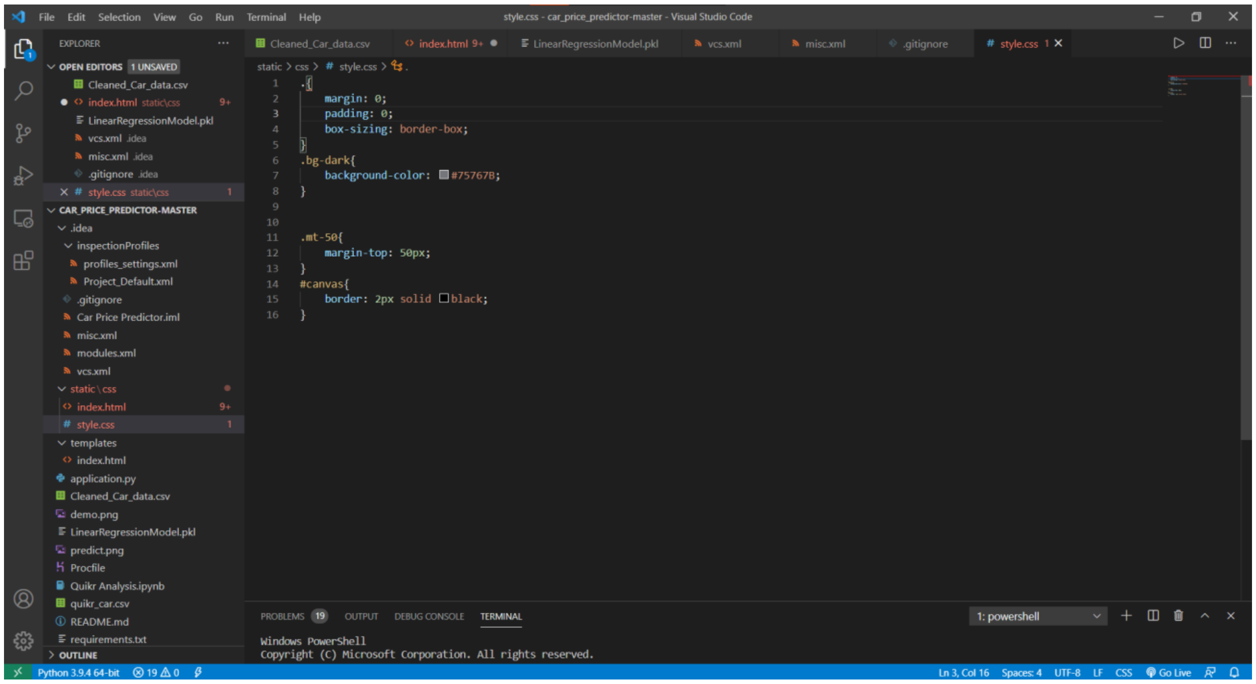
```
In [ ]:
```

VS Code(For User Interface)



```
<!DOCTYPE html>
<html lang="en">
</head>
<body class="bg-dark">

<div class="container">
    <div class="row">
        <div class="card mt-50" style="width: 100%; height: 100%">
            <div class="card-header" style="text-align: center">
                <h1>Welcome to Car Price Predictor</h1>
            </div>
            <div class="card-body">
                <div class="col-12" style="text-align: center">
                    <h5>This app predicts the price of a car you want to sell. Try filling the details below: </h5>
                </div>
                <br>
                <form method="post" accept-charset="utf-8" name="Modelform">
                    <div class="col-md-10 form-group" style="text-align: center">
                        <label><b>Select the company:</b> </label><br>
                        <select class="selectpicker form-control" id="company" name="company" required="1"
                                onchange="load_car_models(this.id,'car_models')">
                            {% for company in companies %}
                            <option value="{{ company }}">{{ company }}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <div class="col-md-10 form-group" style="text-align: center">
                        <label><b>Select the model:</b> </label><br>
                        <select class="selectpicker form-control" id="car_models" name="car_models" required="1">
                        </select>
                    </div>
                    <div class="col-md-10 form-group" style="text-align: center">
                        <label><b>Select Year of Purchase:</b> </label><br>
                        <select class="selectpicker form-control" id="year" name="year" required="1">
```

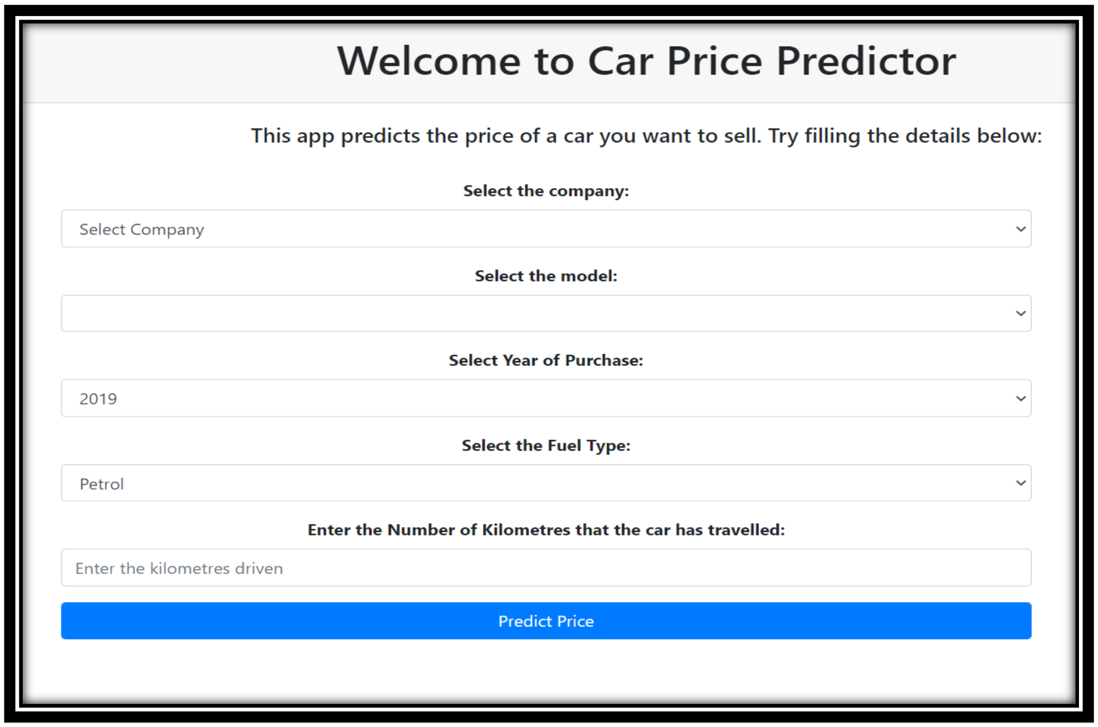
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```



```
.{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
.bg-dark{
    background-color: #75767B;
}


.mt-50{
    margin-top: 50px;
}
#canvas{
    border: 2px solid black;
}
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

Test Result Output Screenshot:



Welcome to Car Price Predictor

This app predicts the price of a car you want to sell. Try filling the details below:

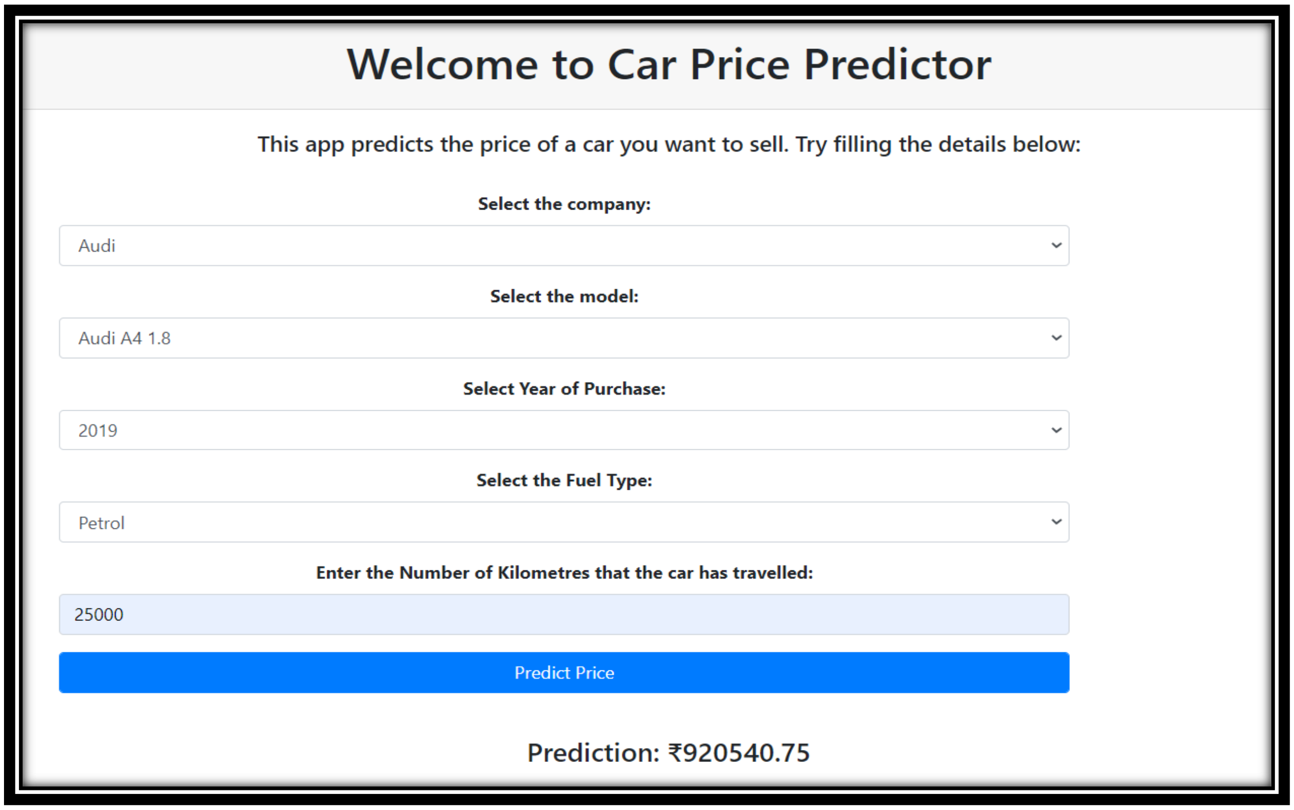
Select the company: Select Company

Select the model:

Select Year of Purchase: 2019

Select the Fuel Type: Petrol

Enter the Number of Kilometres that the car has travelled: Enter the kilometres driven



Welcome to Car Price Predictor

This app predicts the price of a car you want to sell. Try filling the details below:

Select the company: Audi

Select the model: Audi A4 1.8

Select Year of Purchase: 2019

Select the Fuel Type: Petrol

Enter the Number of Kilometres that the car has travelled: 25000

Prediction: ₹920540.75

Reference:

- First of all the data was scraped from Quikr.com (https://quikr.com)
- Kaggle.com (Dataset)
- Google.com
- My Github Link (https://github.com/HumayunKABIR-HK/Artificial-Intelligence-Machine-Learning-Project.git)

Conclusion:

Vehicle price prediction can be a challenging task due to the more numbers of attributes that should be considered for the accurate prediction. The collection and preprocessing of data is the major step in prediction. In this paper, to normalize, standardize, and clean the data, PHP scripts were built. This will be used to avoid unnecessary noise for machine learning algorithms.

The prediction performance must be increased by using data cleaning processes. But in this paper, the insufficient set of complex data is the drawback here. We will get only 50 percent result on applying the single machine algorithm.

Therefore, we proposed multiple groups of machine learning algorithms to gain more accuracy and it achieved 93 percent of efficiency. This comparison of single and multiple groups of the machine learning algorithms is significant. And also, it overcomes the drawback of the single machine algorithm which is given in the proposed system. Although, this system has achieved valuable performance in vehicle price prediction, our aim for future work is to test this system to work successfully with various data sets.

Thank you