What would be the price of future cars?

Section 1: Project Definition

Project Overview

In 2022, 67.2 million cars were sold globally. For an automotive company to sell as many cars as possible, an important consideration is the price of their cars. Finding accurate car pricing, based on historical trends and competitors, plays an essential role which has the potential to increase sales.

This project will aim to predict the price of cars based on historical data found in Kaggle (https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data). The dataset contains data from Craigslist on used vehicles for sale, containing 26 features and 426, 880 rows as seen in Figure 1. The Figure below also shows the column names where price is the target variable.

```
df.shape
```

```
(426880, 26)
```

```
df.columns
```

```
Index(['id', 'url', 'region', 'region_url', 'price', 'year', 'manufacturer',
       'model', 'condition', 'cylinders', 'fuel', 'odometer', 'title_status',
       'transmission', 'VIN', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'description', 'county', 'state', 'lat', 'long',
       'posting_date'],
      dtype='object')
```

Figure 1: Data summary

Problem Statement

By utilising historical data and a number features from Figure 1, the following project will undergo a methodology to predict the price of cars.

This will be done by analysing the data using visualisations and statistics; cleaning the dataset by identifying missing values and outliers; conducting feature engineering to simplify features; splitting the data into train and test; running a Random Forest Regressor; improving the model parameters through hyper-parameter tuning and finally discussing the results.

The aim is to achieve an R2 score of at least 70%.

Metrics

Since this is a regression problem, the main metric that will be used in the project is R2 score. The R2 score is a measurement of how well the regression predictions approximate the real data points. In other words, the closer the predicted data points are to the real data points, the higher the R2 score. Therefore, when predicting against a continuous target variable such as price , the R2 will help identify how well the data fit the regression line.

Section 2: Analysis

Data Exploration

Data exploration began with a general overview of the data. In Figure 2, information is shown about the columns, highlighting the number of entries, missing values and column data types. It appears that there are a number of missing values in most columns, since the number of entries varies considerably between the features and this will be assessed later. The feature 'County' specifically stands out as there are no data entries in the column. Another important observation is the different data types present. For a regression based model, the algorithm works with floating or integer data types only, so the 19 object columns will need to be encoded.

Figure 3 dives deeper into missing values. Only 6 out of the 26 columns have no missing values whereas 7 columns have more than 100,000 rows of missing data. Missing values will be dealt when the data is preprocessed.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 26 columns):
id              426880 non-null int64
url             426880 non-null object
region          426880 non-null object
region_url      426880 non-null object
price           426880 non-null int64
year            425675 non-null float64
manufacturer    409234 non-null object
model           421603 non-null object
condition       252776 non-null object
cylinders       249202 non-null object
fuel            423867 non-null object
odometer        422480 non-null float64
title_status    418638 non-null object
transmission    424324 non-null object
VIN             265838 non-null object
drive           296313 non-null object
size            120519 non-null object
type            334022 non-null object
paint_color     296677 non-null object
image_url       426812 non-null object
description     426810 non-null object
county          0 non-null float64
state           426880 non-null object
lat             420331 non-null float64
long            420331 non-null float64
posting_date    426812 non-null object
dtypes: float64(5), int64(2), object(19)
memory usage: 84.7+ MB
```

Figure 2: Column information

```
id                  0
url                 0
region              0
region_url          0
price               0
year             1205
manufacturer    17646
model            5277
condition      174104
cylinders      177678
fuel             3013
odometer         4400
title_status     8242
transmission     2556
VIN            161042
drive          130567
size           306361
type            92858
paint_color    130203
image_url          68
description        70
county         426880
state               0
lat              6549
long             6549
posting_date       68
dtype: int64
```

Figure 3: Count of Null values

Next, the statistics of each column was looked at, shown in Figure 4. Again, county having no values is reassured but more interestingly are the price and odometer features. The maximum values for both are significantly larger than their minimum values and are also larger than the 50th and 75th quartile value. This is further backed up by the large standard deviation value and suggests that there may be outliers present in both columns. This will therefore be further looked at in the data visualisation section.

	id	price	year	odometer	county	lat	long
count	4.268800e+05	4.268800e+05	425675.000000	4.224800e+05	0.0	420331.000000	420331.000000
mean	7.311487e+09	7.519903e+04	2011.235191	9.804333e+04	NaN	38.493940	-94.748599
std	4.473170e+06	1.218228e+07	9.452120	2.138815e+05	NaN	5.841533	18.365462
min	7.207408e+09	0.000000e+00	1900.000000	0.000000e+00	NaN	-84.122245	-159.827728
25%	7.308143e+09	5.900000e+03	2008.000000	3.770400e+04	NaN	34.601900	-111.939847
50%	7.312621e+09	1.395000e+04	2013.000000	8.554800e+04	NaN	39.150100	-88.432600
75%	7.315254e+09	2.648575e+04	2017.000000	1.335425e+05	NaN	42.398900	-80.832039
max	7.317101e+09	3.736929e+09	2022.000000	1.000000e+07	NaN	82.390818	173.885502

Figure 4: Column statistics

Finally, duplicate values were checked. From Figure 5, only 39 duplicated rows are found and can be dealt with easily during preprocessing

```
# Checking for duplicates

df.duplicated().sum()
```

```
39
```

Figure 5: Duplicate rows

Data Visualisation

It was identified there may be a possibility for outliers in price and odometer columns. To further evaluate, count plots for price and odometer are shown in Figures 6 and 7, respectively. Starting with price, the data is heavily skewed to the left where 2 data points are identified as outliers around the 1.2e9 an 3.8e9 prices. A similar case is seen for odometer where, again, 2 data points are heavily skewing the plot to the left. Interestingly, these are occurring at the same price ranges suggesting that a filter is needed for the price column which will be added during preprocessing.

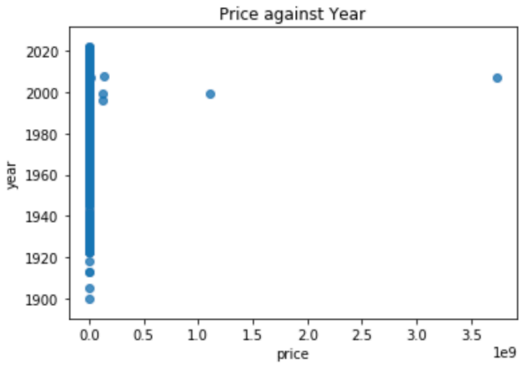

Figure 6: Price distribution

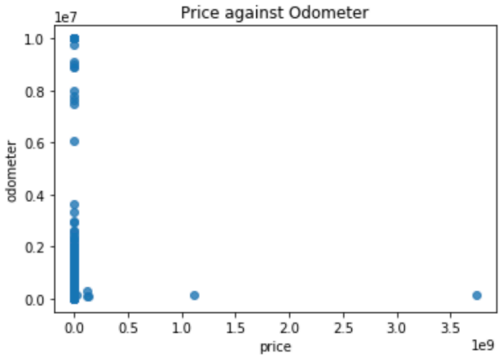

Figure 7: Odometer distribution

To understand the data further, distributions for manufacturers and vehicle type are seen in Figures 8 and 9, respectively. In terms of manufacturers, Ford with an automatic gearbox is the most popular, significantly outpacing the others. Chevrolet with an automatic gearbox and Toyota with an automatic gearbox follow. It is also interesting to note that an automatic transmissions hugely popular compared to the others, in fact, the count for all manufacturers is dominated by automatic gearboxes.

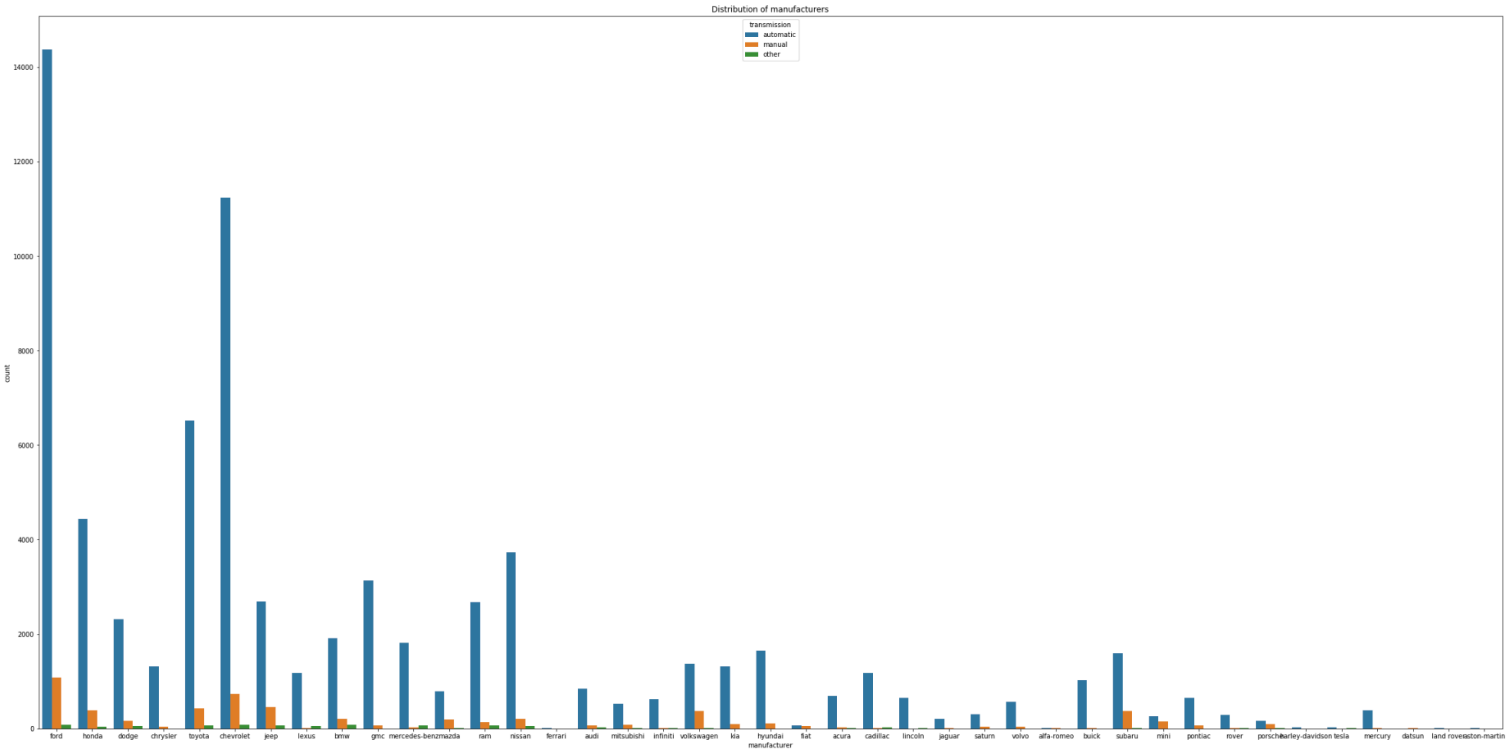

Figure 8: Distribution of manufacturers

Next is the distribution of vehicle type. From Figure 9, Sedans with front wheel drive narrowly edge out on top, followed by SUV's with 4 wheel drive system. Unlike the dominance of automatic transmissions, the drive system of cars are dependent on vehicle type. 5 vehicle types are popular with 4WD, 3 are popular with FWD and 5 are popular with RWD.

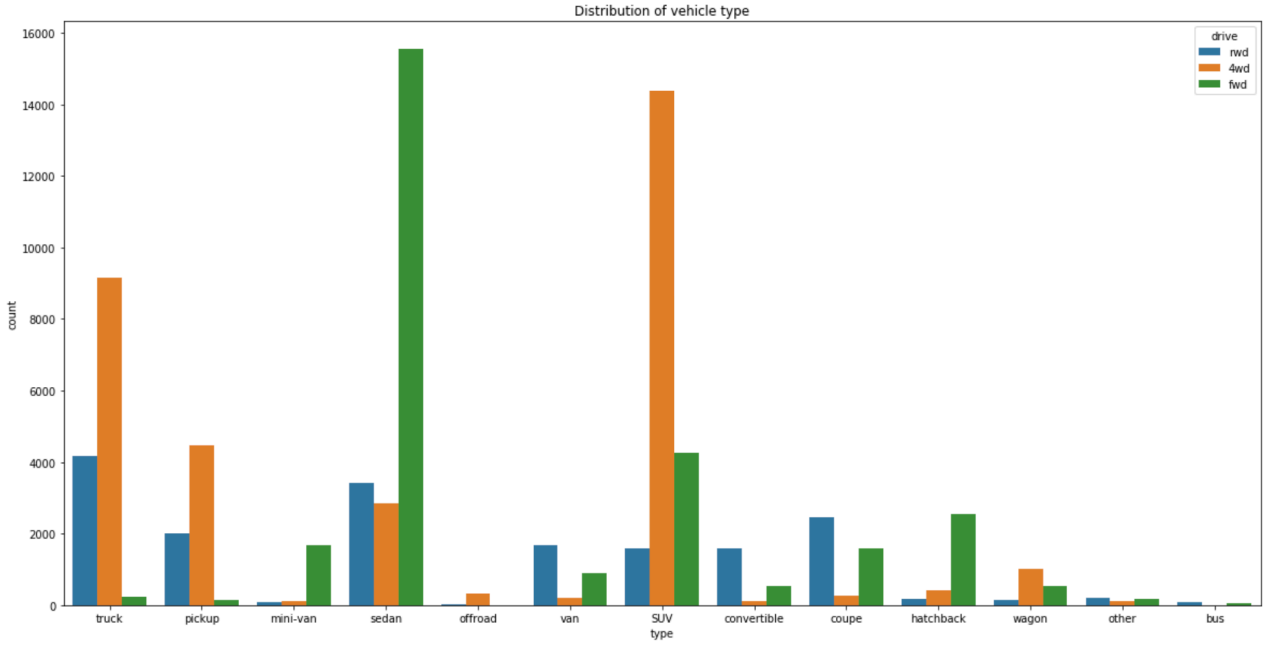

Figure 9: Distribution of vehicle type

Section 3: Methodology

Data Preprocessing

All the changes mentioned in the data analysis and data exploration sections were addressed during preprocessing. Firstly, columns that were not needed or had little contribution to price were dropped as seen in Figure 10. These were done manually or through research to identify any redundant columns.

To deal with Null values, the majority were removed when dropping unnecessary columns from Figure 10. However, the remaining Null values were removed by dropping the columns.

```
# Dropping columns not needed or with high Null values

df.drop(['id', 'url', 'region_url', 'VIN', 'image_url', 'description', 'county', 'lat',
         'long'], axis = 1, inplace = True)
```

Figure 10: Columns to drop

To access the issue of duplicate values, only the first duplicated row was kept and the rest removed, shown in the Figure 11.

```
# Dropping duplicates

df.drop_duplicates(keep='first', inplace=True)
```

Figure 11: Dropping duplicated values

The next issue was the problem of outliers in the price column. It was found that the outliers were only an issue when price exceeded 1,000,000, therefore, the data was filtered for values below that value. This is shown in Figure 12.

```
# Dealing with outliers

df2 = df1[df1['price'] < 1e6]
df2.head()
```

Figure 12: Filtering outliers

To check the distributions after the outliers are removed, the same plots from the data visualisation section are plotted in Figures 13 and 14. Now one can see that the majority of newer cars are generally priced less and cars that are more expensive tend to have lower mileage.

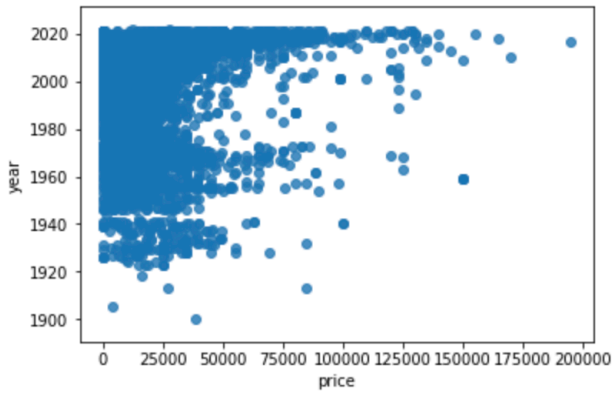

Figure 13: Year against Price

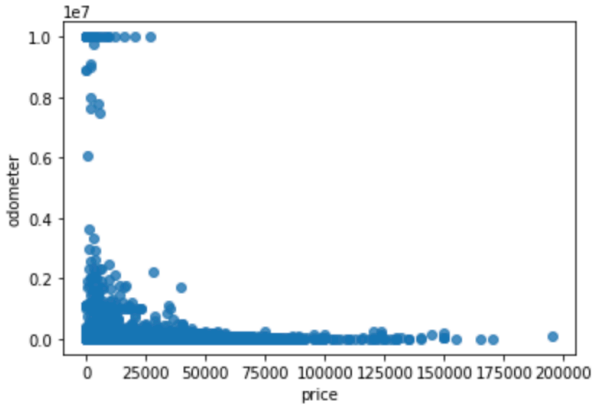

Figure 14: Odometer against Price

The final change required during preprocessing was dealing with the differing data types of the columns. This was solved simply by using categorical variable encoding, as seen in Figure 15. Variable encoding converts categorical features into separate features where the values for each columns are integers.

```
# Categorical variable encoding

from sklearn.preprocessing import LabelEncoder
cat_features = ['region', 'manufacturer', 'model', 'condition', 'cylinders', 'fuel', 'title_status', 'transmission',
                'drive', 'size', 'type', 'paint_color', 'state', 'posting_date']
encoder = LabelEncoder()
encoded = df2[cat_features].apply(encoder.fit_transform)
df2.drop(cat_features, axis=1, inplace=True)
df2 = pd.concat([encoded,df2],axis = 1)
df2.head(1)
```

Figure 15: Categorical variable encoding

After implementing all the preprocessing steps, the finalised data set is seen in Figure 16. By removing unnecessary columns and missing values, the count has reduced to 79,184 and is now equal amongst all the columns. The datatypes are now integers or floats and ready for data modelling.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 79184 entries, 31 to 426836
Data columns (total 17 columns):
region          79184 non-null int64
manufacturer    79184 non-null int64
model           79184 non-null int64
condition       79184 non-null int64
cylinders       79184 non-null int64
fuel            79184 non-null int64
title_status    79184 non-null int64
transmission    79184 non-null int64
drive           79184 non-null int64
size            79184 non-null int64
type            79184 non-null int64
paint_color     79184 non-null int64
state           79184 non-null int64
posting_date    79184 non-null int64
price           79184 non-null int64
year            79184 non-null float64
odometer        79184 non-null float64
dtypes: float64(2), int64(15)
memory usage: 10.9 MB
```

Figure 16: Summary of dataset

Implementation

The cleaned dataset was split into a training set and a testing set. This is so the data can be modelling on the training set, implemented on the testing set and evaluated against both. In Figure 17, the X variables are the features used to predict the y column which in this case in the price.

```
X = df2[['region', 'manufacturer', 'model', 'condition', 'cylinders', 'fuel', 'title_status', 'transmission',
        'drive', 'size', 'type', 'paint_color', 'state', 'posting_date', 'year', 'odometer']]

y = df2['price']
```

Figure 17: Train test split

For the modelling, a decision tree regressor was used. A decision tree was chosen as the method for the model due to its simplicity and low computational efficiency. Since the problem consisted of a continuous target variable, where the price can in theory be any value, a regression model was used since it is best suited for this problem.

Refinement

Once the initial model was run, the number of estimators was 50. However, to optimise the results, hyper parameter tuning was conducted. Firstly the parameters were identified, which was just number of estimators to help limit the run time of the model. GridSearchCV was used as the method since it was a more structured approach, assessing the parameters in a step by step process. Two hyper parameter models were run, each aiming to increase the number of estimators, from 50, 100, 200 to 200, 300 and 400 as seen in Figures 18 and 19, respectively.

```
param_grid = {'n_estimators': [50, 100, 200]}
model = RandomForestRegressor(random_state=0)
grid = GridSearchCV(estimator=model, param_grid=param_grid)
```

Figure 18: First hyper parameter tuning

```
param_grid = {'n_estimators': [200, 300, 400]}
model = RandomForestRegressor(random_state=0)
grid = GridSearchCV(estimator=model, param_grid=param_grid)
```

Figure 19: Second hyper parameter tuning

Section 4: Results

Model Evaluation, Validation and Justification

When running the RandomForestRegessor model with 50 estimators, the R2 score was 0.785. This had already achieved the aim of 70% highlighted at the beginning of the project. Regardless, hyper parameter tuning was conducted to ensure this was the most optimum value for the R2 score. Therefore, the number of estimators was increased and the first hyper parameter tuning conducted. During the first iteration, the best estimator was 200 with an R2 score of 0.79. During the second iteration, the best estimator was 400 with an R2 score of 0.79. Interestingly the R2 score when using an estimator of 200 and 400 were identical as seen in the plot in Figure 20. Therefore, the most optimum number of estimator is 200 since anything higher does not change the value of the R2 score but does increase the complexity and running time of the model.

The final metric and parameter for the model was: R2 score of 0.79 with 200 estimators.

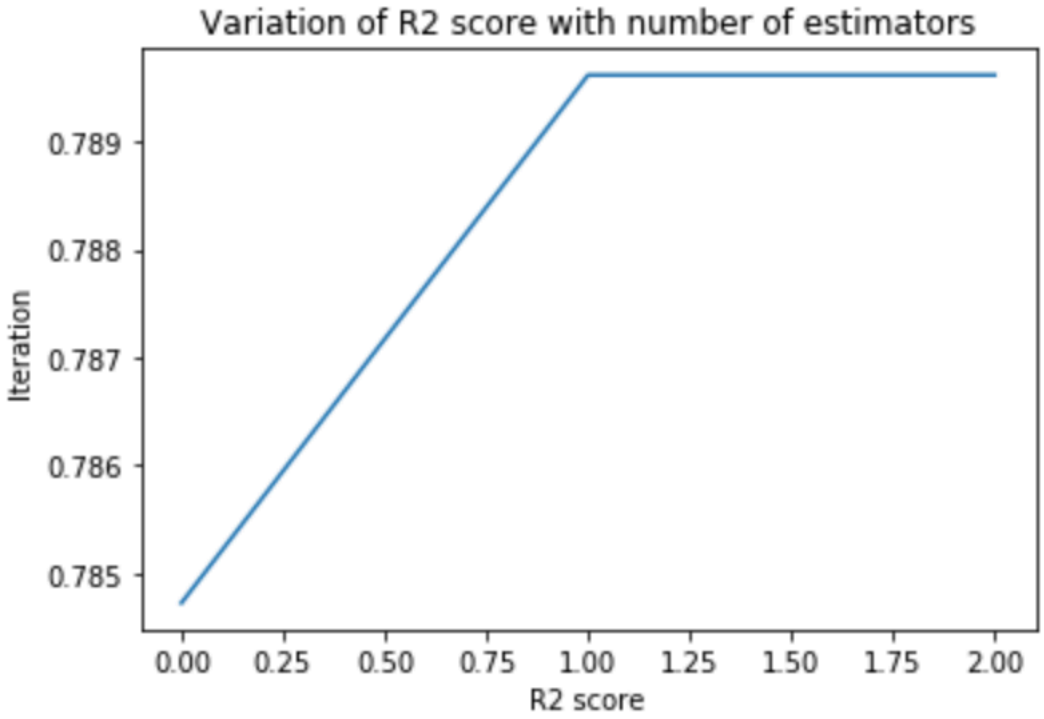

Figure 20: R2 score against tuning iterations

Section 5: Conclusion

Reflection

The purpose of the project was to predict the price of future cars using historical data and several features. The data was first inputted then explored to identify any problems or changes that were required. This was further backed up by the data visualisation phase which helped identify trends and outliers. Next, the data was preprocessed to fix the issues identified and led to the modelling phase. In this section, the data was split, trained and tuned using number of estimators as the parameters in order to achieve an optimum R2 score.

It was interesting cleaning the data since it required different methods and techniques to tackle the issues. There were also multiple ways of solving the issues which made it interesting to find the most suitable or easier way.

Improvement

To improve the project, several different models could be used to predict car prices. For example, a logistic regression model can be compared to a linear regression model which is then compared to the random forest regressor used in the project. The parameters for each model could also be tuned so the optimised results are compared. Then the best model is carried forward in the analysis.