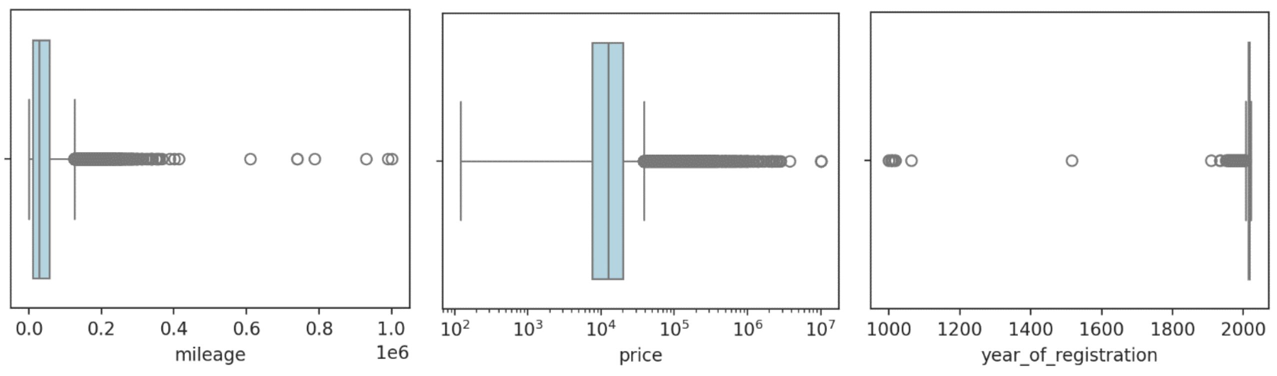
1.0 DATA PROCESSING FOR MACHINE LEARNING

The dataset is an anonymous car adverts data set from Autotrader – an online automotive marketplace, containing information about a car and its selling price. The dataset contains **402005** rows and **12** columns: *public_reference, mileage, reg_code, standard_color, standard_make, standard_model, vehicle_condition, year_of_registration, price,body_type, crossover_car_and_van, fuel_type.*

Detecting Outliers

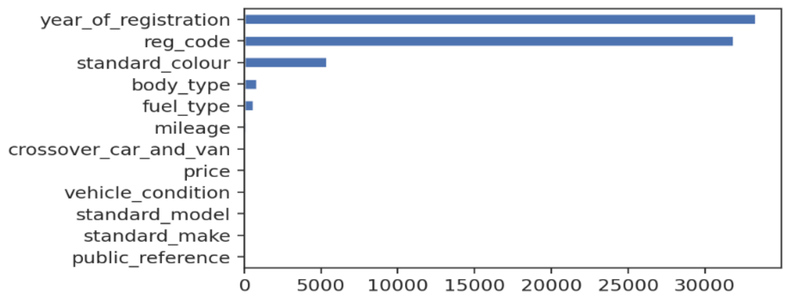
Using a box plot to analyse the distribution of price, mileage and year of registration in order to identify potential outliers.



The mileage has a max value of **999,999 km/hr** and the 75% quartile is **56875.75km/hr**. From the boxplot, there are values above **600,000km/hr**, which are quite unrealistic and could be due to human error. Most of the price data points are lying around **£9,500 - £35,000,** the outliers are due to expensive luxurious cars and human error. The year of registration has values less than **1600** which is clearly from human error and was fixed manually as these values were just a few. To address the outliers, the mileage was capped at **250,000km/hr** because some cars with high mileage could have been taxis. Price values were capped at between **1%** quartile and **99%** quartile to handle the extremely high prices.

Missing Values

Visualising the missing values in the dataset, the year_of_registration column has the highest amount of missing values. The year of registration can be deduced from its reg_code value. There are **1741** rows where reg_code is not null and year_of_registration is null. Scraping the accurate registration code and year identifiers from the UK's Registration code on Wikipedia, I was able to fill in these values. There are **31249** rows where the year_of_registration is null and vehicle condition is 'NEW', I filled this with the max year_of_registration which is **2020,** The public reference column which seems to contain the date the car was advertised also aided in my decision. The other missing values were imputed using *SimpleImputer()* from sci-kit learn after **splitting my data into train and test to avoid data leakage** (mean for numerical, most frequent for categorical and year of registration).



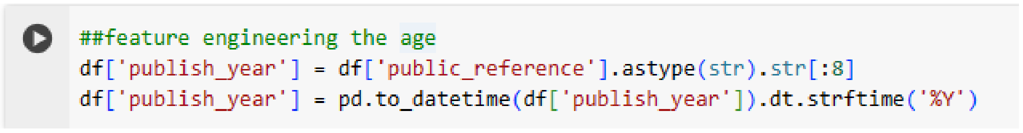
Encoding and Scaling

The Numerical features were scaled using the *StandardScaler()* from sci-kit learn. The Categorical features were Target Encoded and scaled using a *TargetEncoder()* and *Standard Scaler()* respectively. The categorical features were not scaled for tree-based models.

2.0 FEATURE ENGINEERING

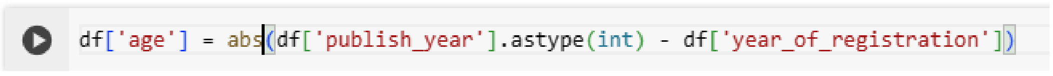
Engineering Age of Car

The public reference is a unique identifier for each row but from close observation, it is noticed that the first 8 digits consist of a date format which I assumed to be the date the advert was published and the remaining digits were the unique identifier. I performed string manipulation to extract the date, furthermore pandas *datetime()* was used to from the year from the already extracted date.



```
##feature engineering the age
df['publish_year'] = df['public_reference'].astype(str).str[:8]
df['publish_year'] = pd.to_datetime(df['publish_year']).dt.strftime('%Y')
```

The car age was deduced by subtracting the year the car was registered from the year the car advert was placed. This is the car age at the time the advert was published.

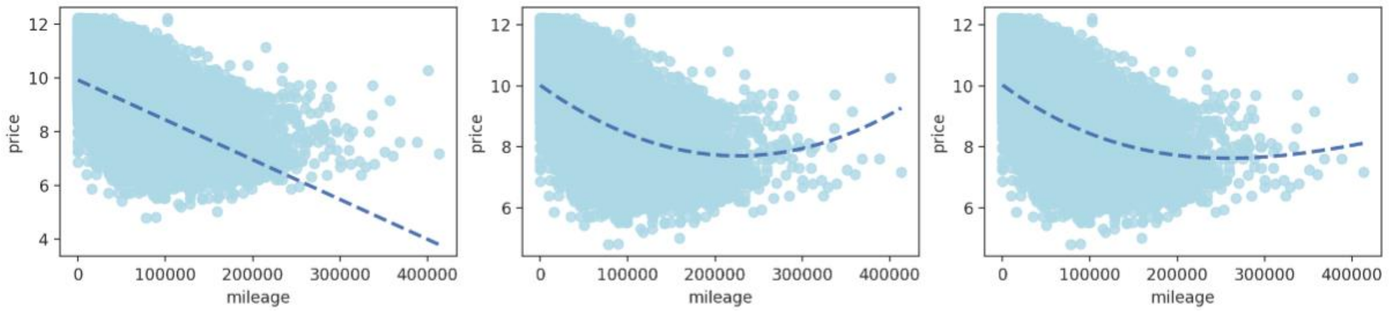


```
df['age'] = abs(df['publish_year'].astype(int) - df['year_of_registration'])
```

The following columns were dropped: *public_reference, reg_code.*

Generating Polynomial Features

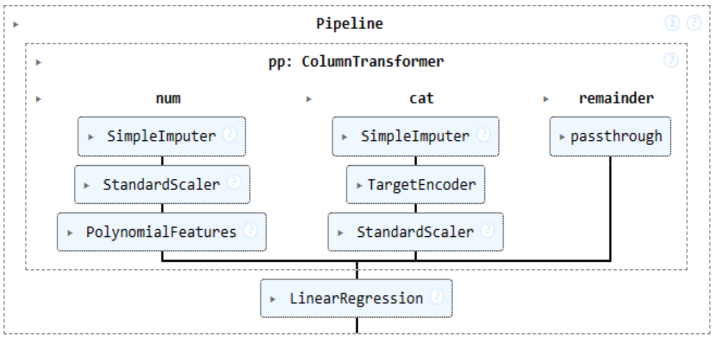
Performing Linear Regression on Price, it can be seen that the line under fits the data and will likely not capture patterns. As we increase the order-which is the degree of polynomial, the line tends to capture more patterns in the data.



Polynomial Features and Interactive Features were generated using sci-kit learn *PolynomialFeatures()* with a degree of **2.** The transformed dataset now contains **14 columns.** The new columns generated are: *mileage^2, mileage_crossover_car_and_van, mileage_age, crossover_car_and_van^2, crossover_car_and_van_age, age^2.*

Comparing Linear Regression with and without polynomial features

Fitting a Linear Regression model with and without Polynomials Features, it is observed that the Model with Polynomial Features performs slightly better with a mean cross-validation score of **0.79** while that without Polynomial Features has a score of **0.78**

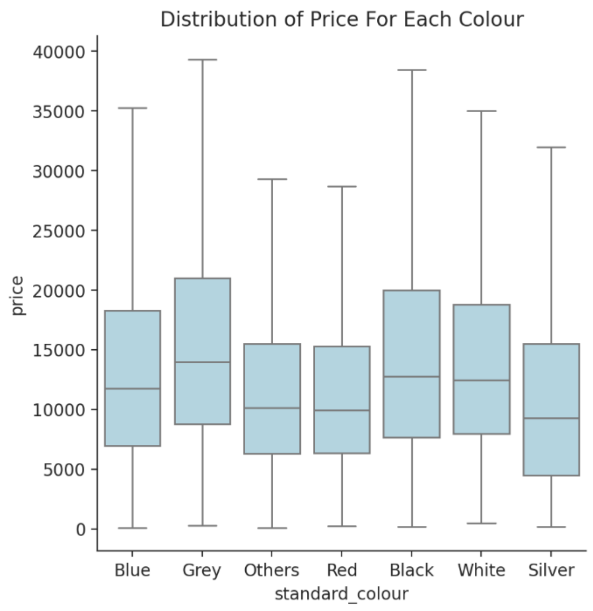


Preprocessing Pipeline

This indicated that the polynomial features does not having an effect on the model and its predictions. Moving forward, manual and automated features will be performed to select the best features from the original **without the polynomial features.**

3.0 FEATURE SELECTION AND DIMENSIONALITY REDUCTION

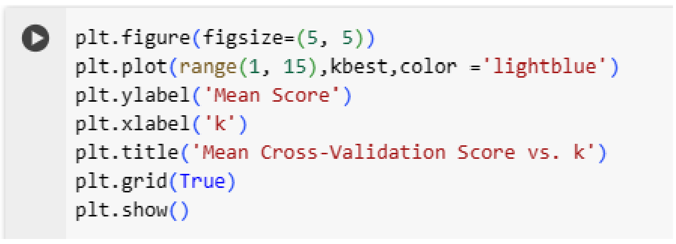
Through EDA(Exploratory Data Analysis) I was able to carryout a manual feature selection by visualising the distribution of price within groups of each feature. The **standard_colour column** was dropped because using a catplot it can be seen that the average price for each colour of car is within the same range and thus the colour of a car does not affect the price of the car.



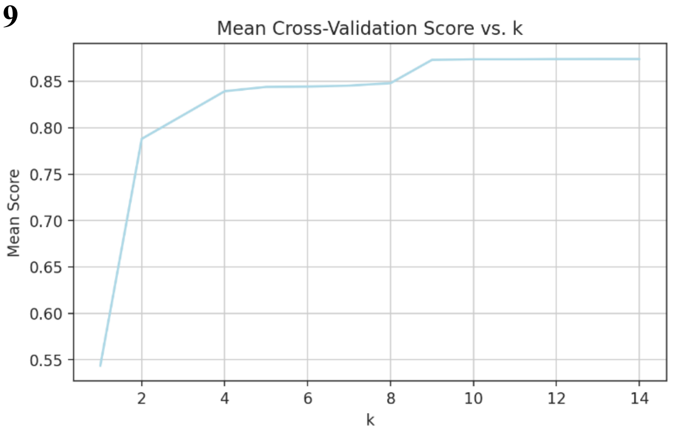
Automated Feature Selection

Automated feature selection was performed using *SelectKBest(), RFECV()* – Recursive Feature Selection, and *SequentialFeatureSelector()* from sci-kit learn library. All methods were tested with Linear Regression and all returned a cross-validation score within 0.86 – 0.87.

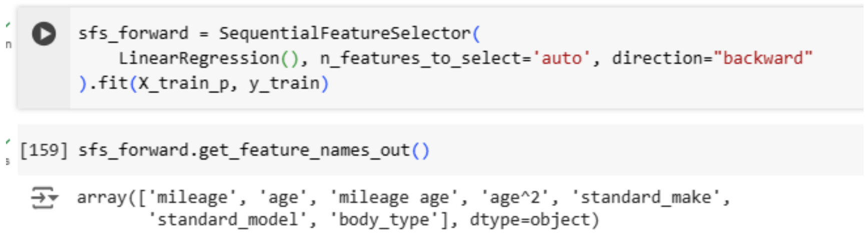
Calculating and plotting cross-validation scores for when **k = 1 to the total Number of Features** it is observed that cross-validation scores flatten out at **k = 9**



```
plt.figure(figsize=(5, 5))
plt.plot(range(1, 15),kbest,color ='lightblue')
plt.ylabel('Mean Score')
plt.xlabel('k')
plt.title('Mean Cross-Validation Score vs. k')
plt.grid(True)
plt.show()
```



With all selectors having similar cross-validation scores the **trade-off was between the number of features.** *RFECV()* achieved its score with **12 features,** *SequentialFeatureSelector()* achieved its score with 7 features and thus **proceeded** with the 7 features selected by *SequentialFeatureSelector()*.



```
sfs_forward = SequentialFeatureSelector(
    LinearRegression(), n_features_to_select='auto', direction="backward"
).fit(X_train_p, y_train)
```

```
[159] sfs_forward.get_feature_names_out()

array(['mileage', 'age', 'mileage age', 'age^2', 'standard_make',
       'standard_model', 'body_type'], dtype=object)
```

Principal Component Analysis

Reducing the dimensions through principal component analysis and fitting a Linear Regression Model with the transformed dataset and cross-validating, it is observed that the score did not improve.

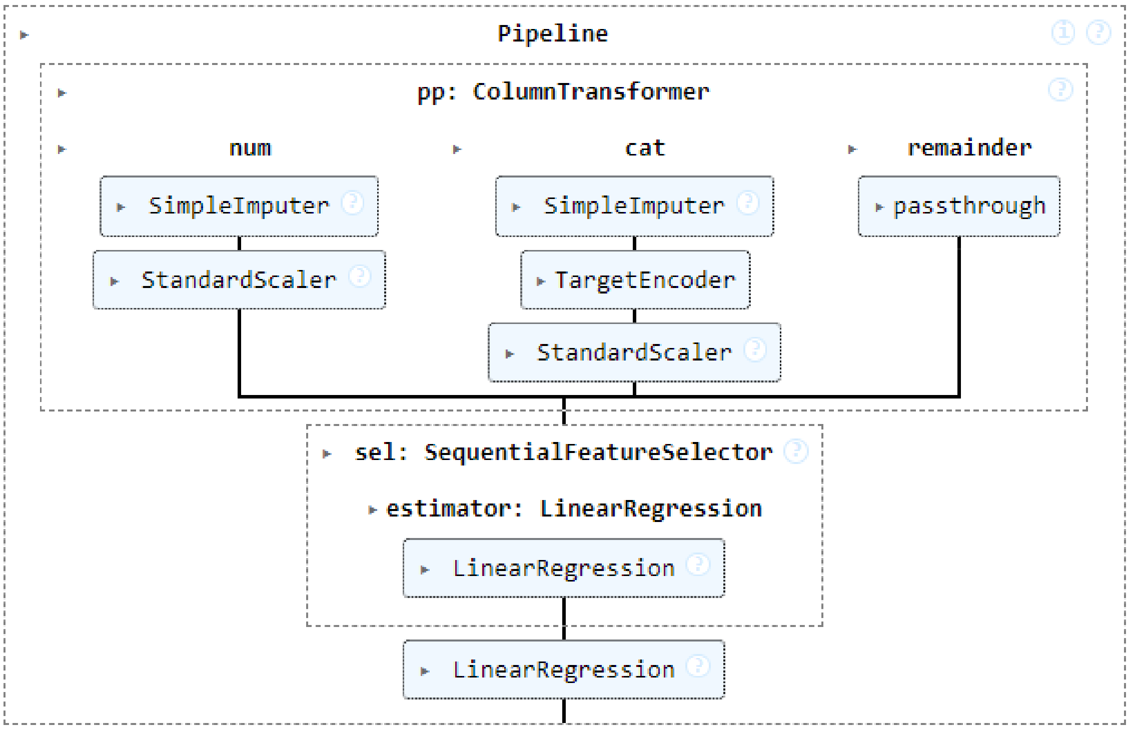
4.0 MODEL BUILDING

Four Machine Learning Models were built: Linear Regression, Random Forest Regressor, Gradient Boosting Regressor and a Voting Regressor(Ensemble). A pipeline was created to preprocess(Impute missing values, encode, scale, and perform feature selection using *SequentialFeatureSelector()*) and fit the models. The sci-kit learn library was used for these processes.

Linear Regression

Linear Regression is used to model the relationship between price(dependent variable) and the other features(independent variables). *LinearRegression*() was fitted with X_train and y_train values through a preprocessing pipeline. The processing pipeline has steps that will:

- Impute missing values for numerical features with their mean values and categorical features with their mode using *SimpleImputer()*
- Scale numerical and categorical features using *StandardScaler()*
- Encode categorical features using *TargetEncoder()*
- Perform automated feature selection using *SequentialFeatureSelector()* with *LinearRegression()* as the estimator



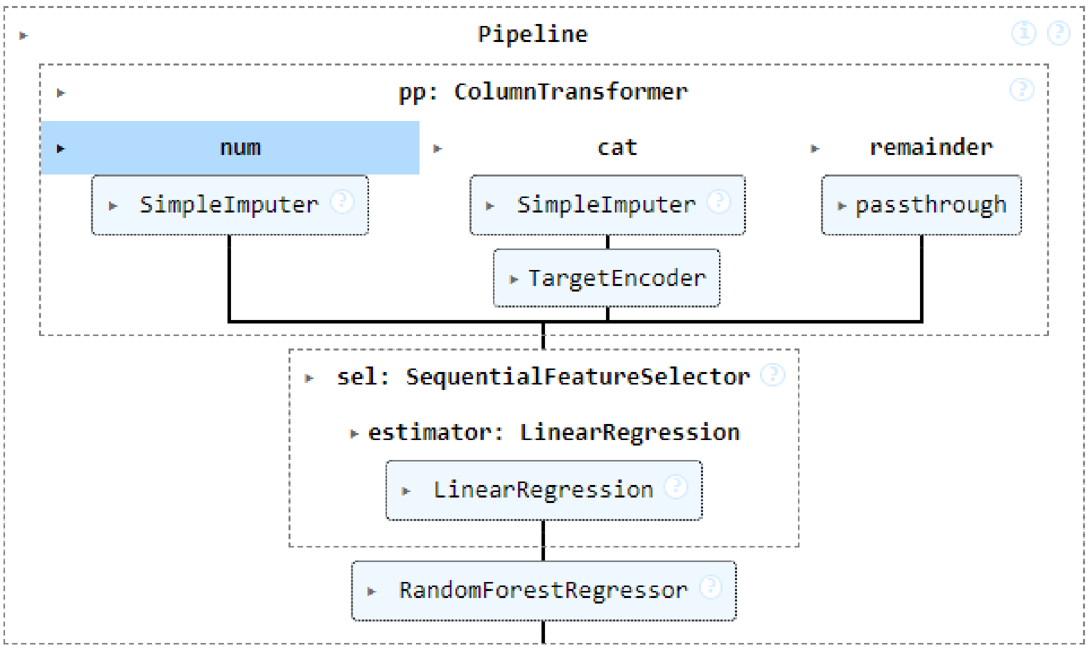
For a Linear Regression model features need to be scaled to ensure that regularization is applied uniformly across all features.

Random Forest Regressor

The Random Forest Regressor is an ensemble technique were the estimations of multiple Decision Trees are averaged and a final prediction is outputted. *RandomForestRegressor()* was fitted with X_train and y_train values through a preprocessing pipeline. The preprocessing pipeline has steps that will

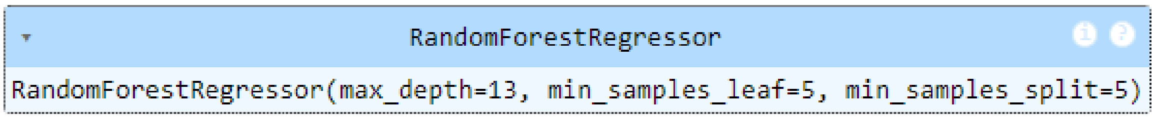
- Impute missing values for numerical features with their mean values and categorical features with their mode using *SimpleImputer()*
- Encode categorical features using *TargetEncoder()*
- Perform automated feature selection using *SequentialFeatureSelector()* with *LinearRegression()* as the estimator

Scaling the features for Random Forest Regressor is not necessary as Tree based models are not sensitive to unregularized data. Scaling will add minimal to no effect on the outcome of models output.



The Random Forest Regressor model has hyperparameters that can tuned. A grid search was initialized for 'max_depth', 'min_samples_split', 'min_samples_leaf'. Estimator was set to 250.

```
gs_results.best_estimator_
```

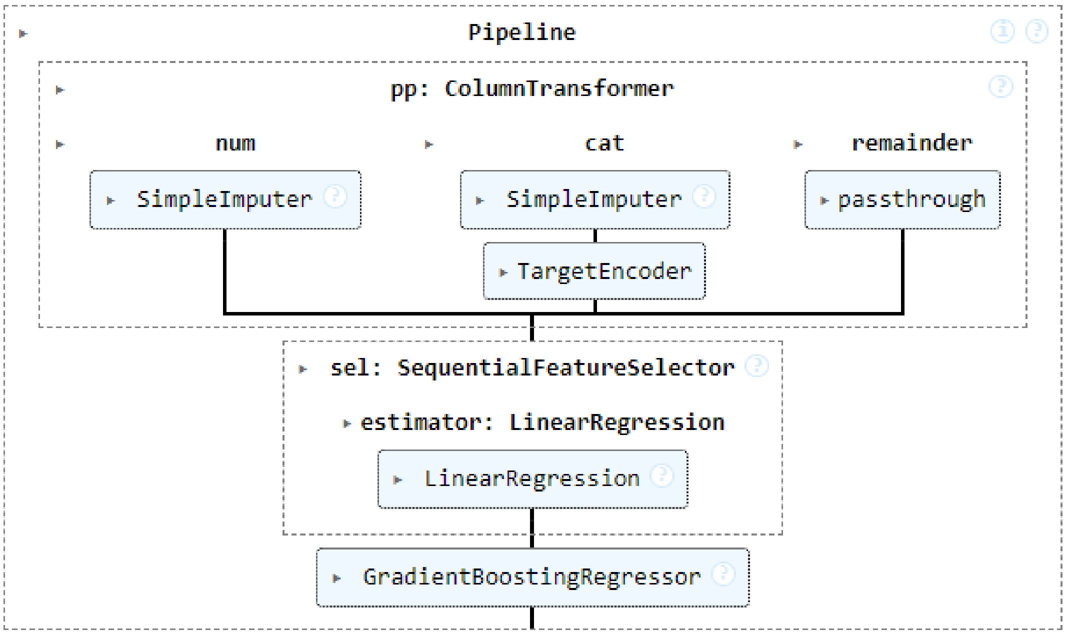


```
RandomForestRegressor
RandomForestRegressor(max_depth=13, min_samples_leaf=5, min_samples_split=5)
```

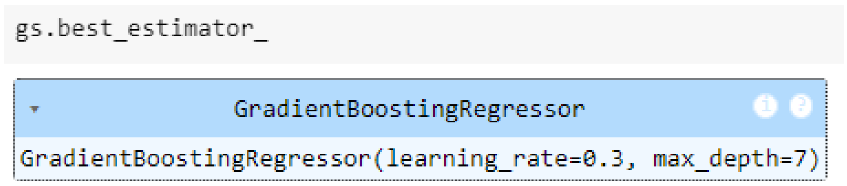
Gradient Boosting Regressor

The Gradient Boosting Regressor is an ensemble technique that builds in a sequential manner where each model tries to learn and correct the errors from the previous models. *GradientBoostingRegressor()* was fitted with X_train and y_train values through a preprocessing pipeline. The preprocessing pipeline has steps that will:

- Impute missing values for numerical features with their mean values and categorical features with their mode using *SimpleImputer()*
- Encode categorical features using *TargetEncoder()*
- Perform automated feature selection using *SequentialFeatureSelector()* with *LinearRegression()* as the estimator



The tree based models is a Tree based model and will not need its features to be scaled because eit has little to no effect on the model's performance. The Gradient Boosting Regressor's hyperparamteres where tuned and estimators set to 250.

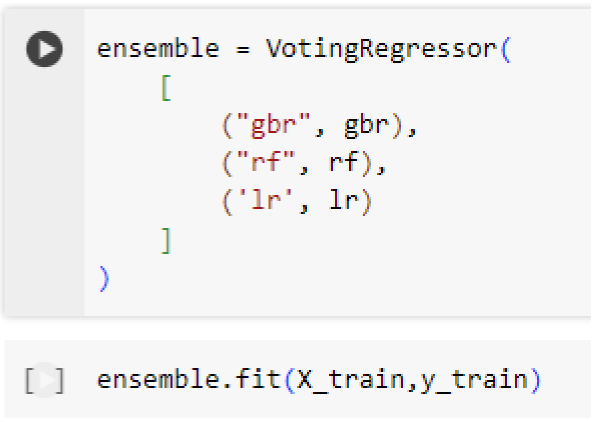


```
gs.best_estimator_
```

```
GradientBoostingRegressor
GradientBoostingRegressor(learning_rate=0.3, max_depth=7)
```

Voting Regressor (Ensemble)

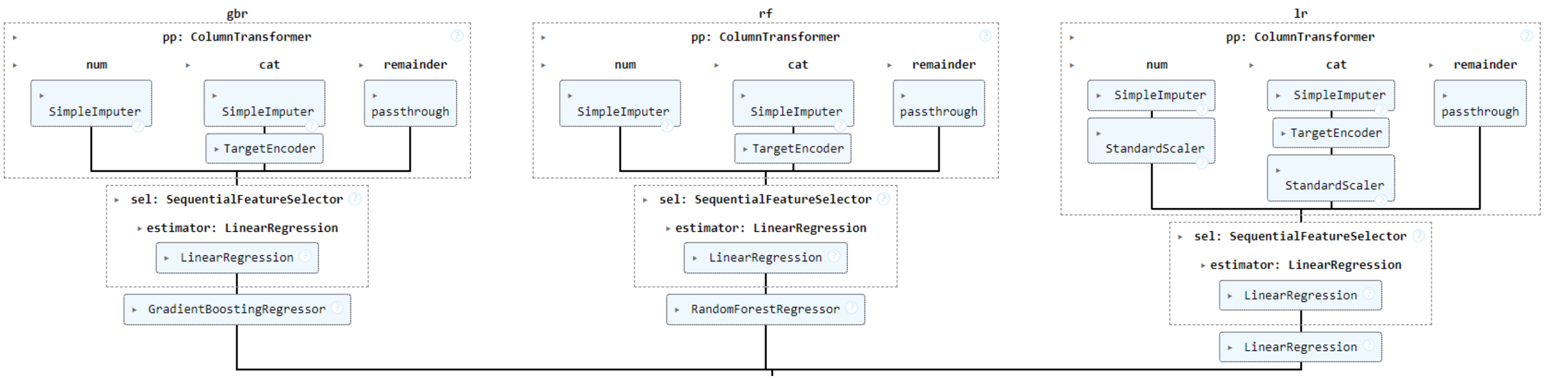
The Voting Regressor is an ensemble technique that combines different regression model to produce a final prediction. Initializing an Ensemble model using *VotingRegressor()* with the three models, Gradient Boosting Regressor, Random Forest Regressor and Linear Regression



```
ensemble = VotingRegressor(
    [
        ("gbr", gbr),
        ("rf", rf),
        ('lr', lr)
    ]
)
```

```
[ ] ensemble.fit(X_train,y_train)
```

Fitting X_train and y_train to the model preprocesses the dataset for each pipeline for each estimator.

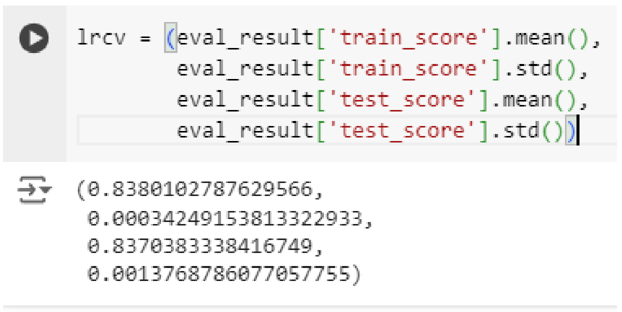


5.0 MODEL EVALUATION AND ANALYSIS

The models will be evaluated by Cross Validating, analysis of True vs Predicted Values, SHAP and PDP plots.

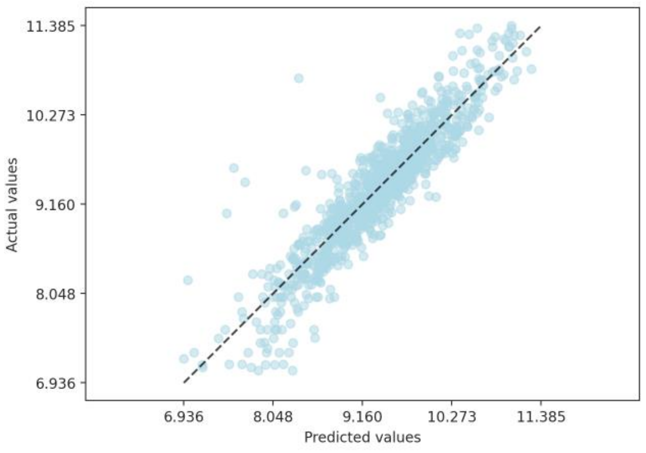
Linear Regression

Cross Validating Linear Regression with folds of 5 we obtain r2 test score of **0.8370** accuracy and standard deviation of 0.00137



```
lrcv = (eval_result['train_score'].mean(),
        eval_result['train_score'].std(),
        eval_result['test_score'].mean(),
        eval_result['test_score'].std())
```

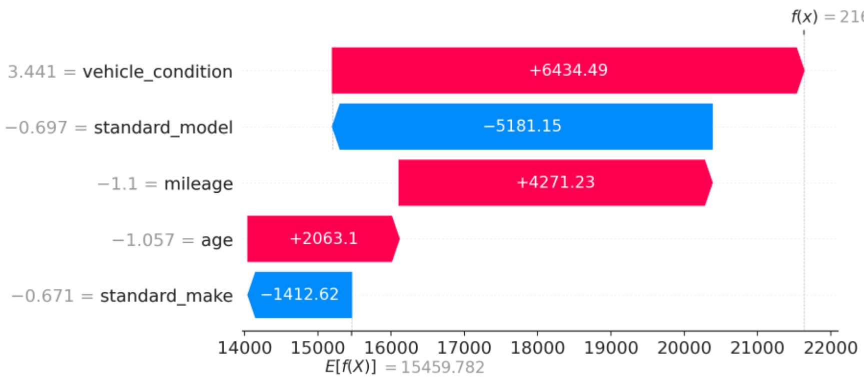
```
(0.8380102787629566,
 0.00034249153813322933,
 0.8370383338416749,
 0.0013768786077057755)
```

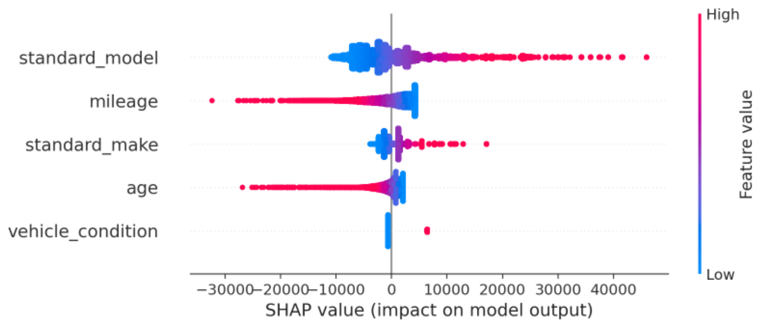


The Linear Regression Model fits the data relatively well but still lacks accuracy that will be required for deployment of model. Some **predicted values** are way off the **true values**.

Analysing with SHAP

Using SHAP to inspect one feature, vehicle_condition has a positive impact on price by **£6434.49.** The standard model has a negative impact on the outcome of price for this feature. The age and make of the car had little impact on the outcome of price. It is highly likely that this car is an old model of a USED car that hasn't been used for long with low mileage.

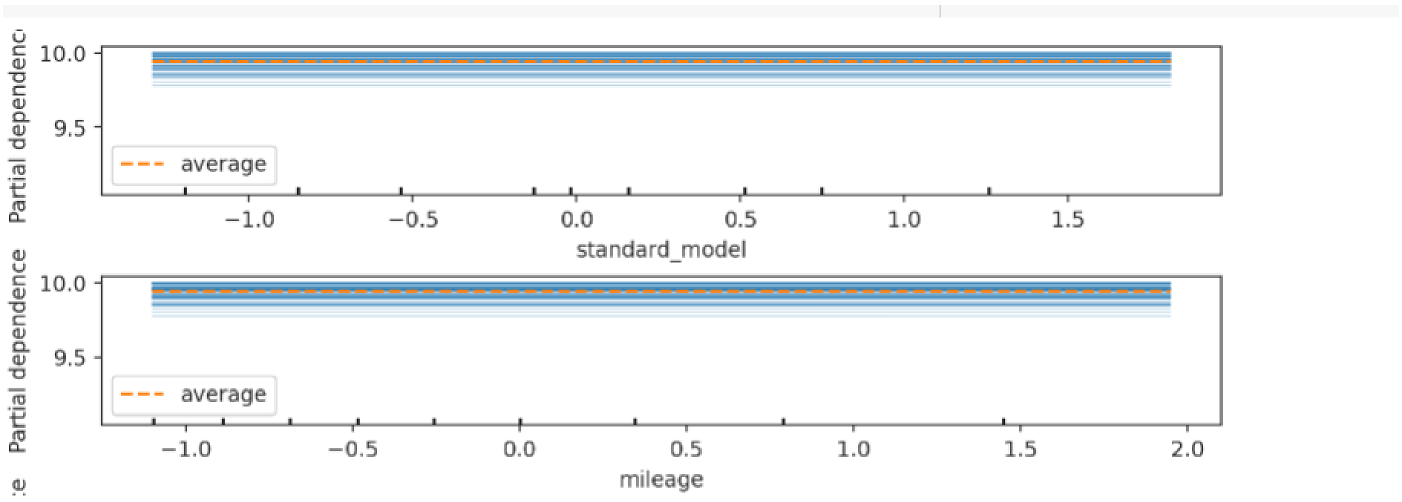




Globally, the standard_model affects the outcome of price positively. The higher the mileage and age the lower the price.

Partial Dependency Plot

The **standard_model** and **mileage appear** to have little effect on price when values are changed in a Linear Regression model

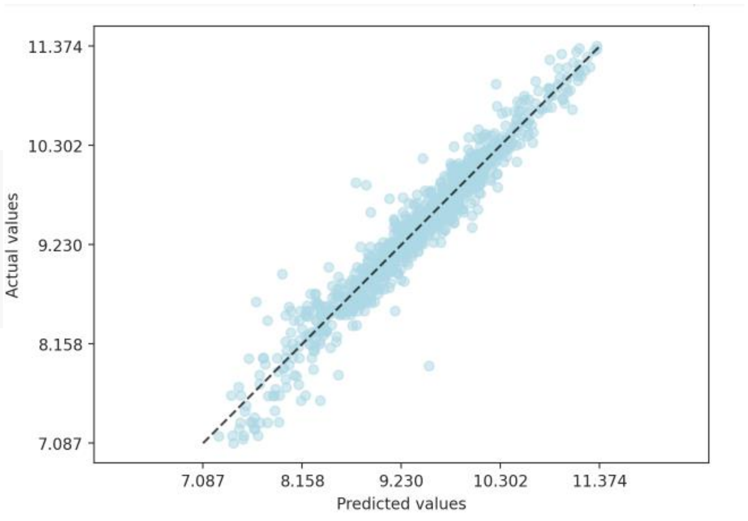


Random Forest Regressor

Cross Validating Random Forest with 5 folds. A high train and test score of **0.95** and **0.94** was achieved respectively.

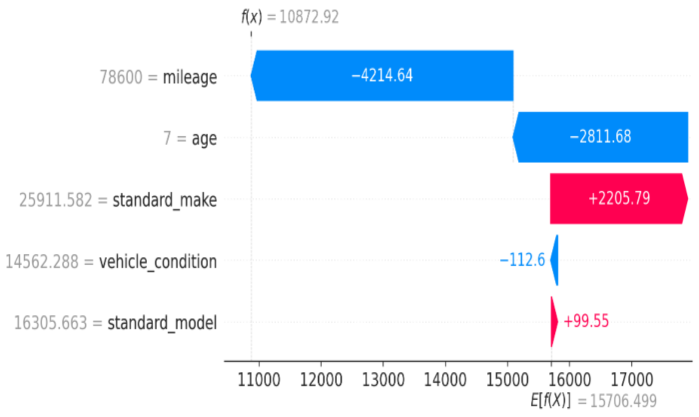
```
eval_result['train_score'].mean(),
eval_result['train_score'].std(),
eval_result['test_score'].mean(),
eval_result['test_score'].std()
```

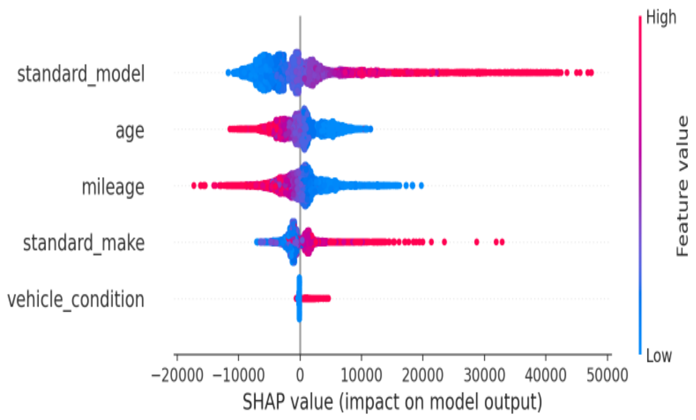
```
(0.9470095395332804,
 0.0008371707673215105,
 0.9379423325766215,
 0.0007174298968073252)
```



Analysing the **True vs Predicted** for Random Forest Regressor Model we observe that the points are more accurate and close to the regression line

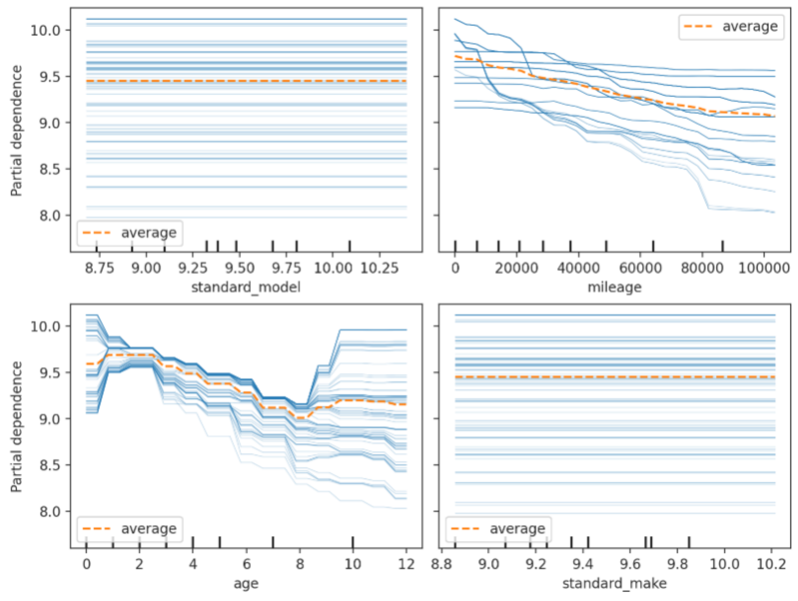
Analysing with SHAP





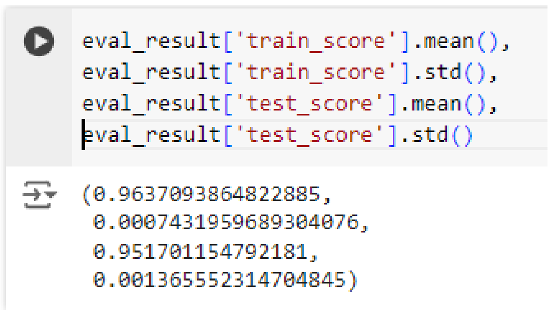
The **Local Explanation** from a random sample indicates that milage and age negatively affects the outcome of price. The Global plot indicates the standard model affects the outcome of price positively. The features are arranged **in increasing order of importance.**

Partial Dependency Plot (Random Forest Regressor



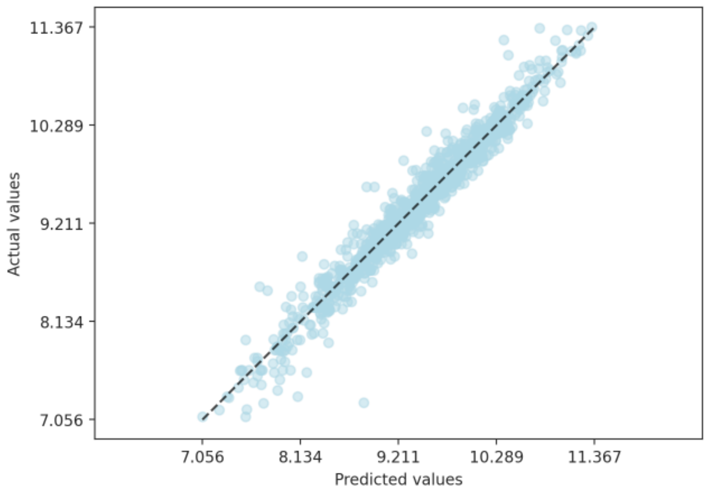
Gradient Boosting Regressor

Cross Validating Gradient Boosting Regressor with 5 folds we obtained a high train and test score of **0.96** and **0.95 respectively**



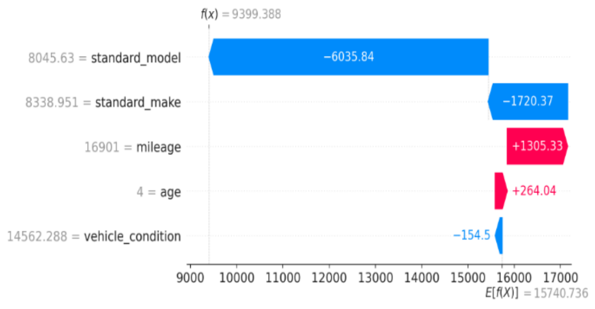
```
eval_result['train_score'].mean(),
eval_result['train_score'].std(),
eval_result['test_score'].mean(),
eval_result['test_score'].std()
```

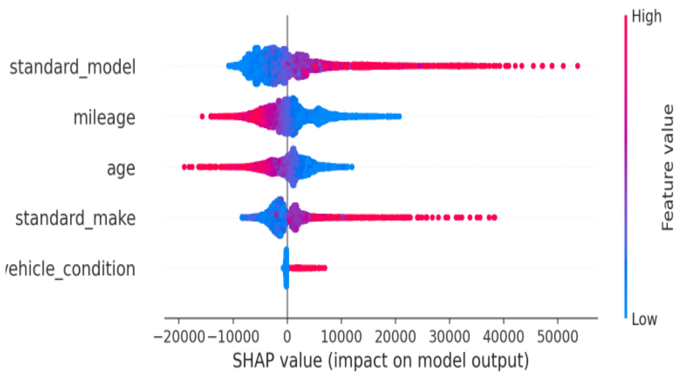
```
(0.9637093864822885,
 0.0007431959689304076,
 0.951701154792181,
 0.0013655523147O4845)
```



Gradient Boosting Regressor does a good job at fitting predicting unseen data as show in the **True vs Predicted**.

Analysing with SHAP

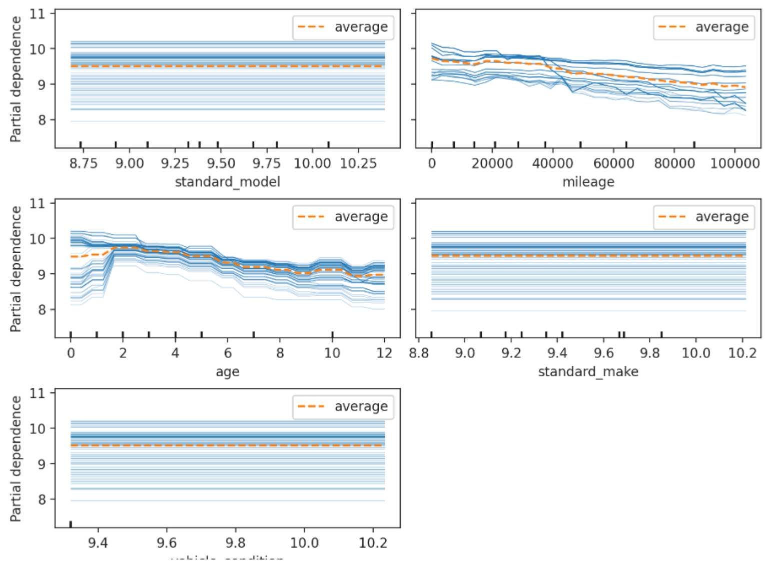




It is observed that standard model is the most impactful feature

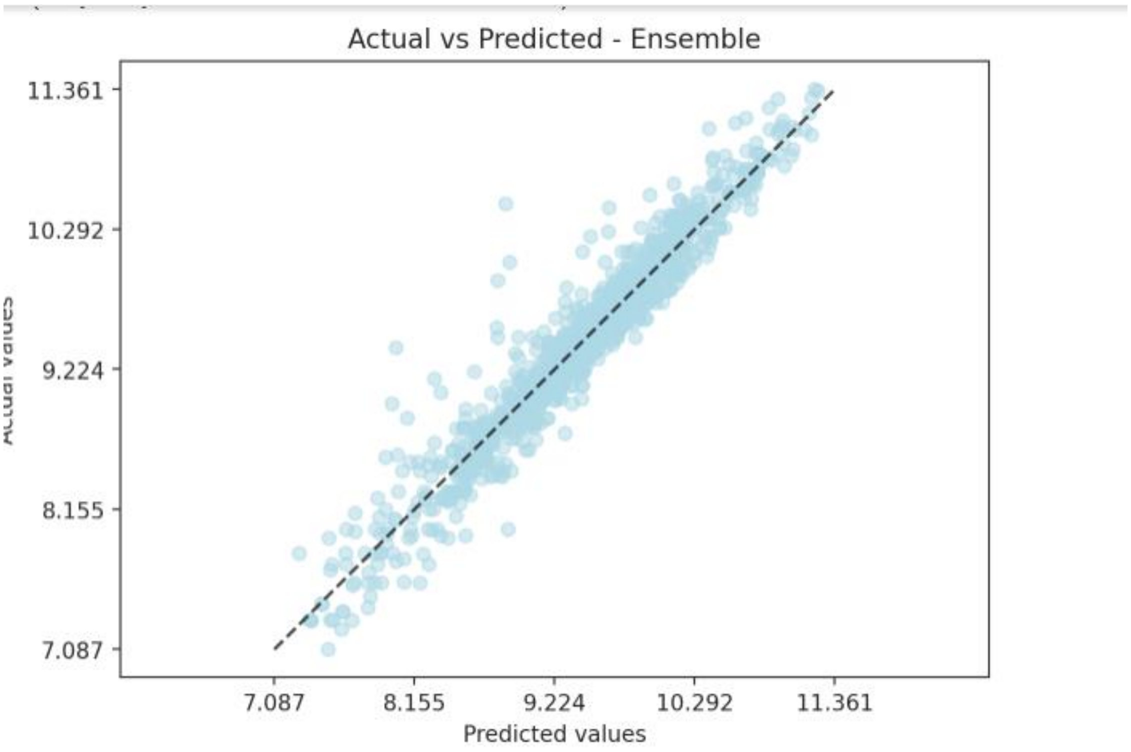
Analysing with Partial Dependency Plot

It is observed that a change in **age and mileage** affects the outcome of price



ENSEMBLE MODEL

The ensemble model returned an R2 score of **0.94.**



In conclusion, the Gradient Boosting Regressor model is the best-fit model for predicting/estimating car prices based on selected inputs.