Supervised Machine Learning: Regression Project

—

Paulina Kossowska
September 2021

“One of the main objectives of this course is to help you gain hands-on experience in communicating insightful and impactful findings to stakeholders. In this project you will use the tools and techniques you learned throughout this course to train a few linear regressions on a data set that you feel passionate about, select the regression that best suits your needs, and communicate insights you found from your modeling exercise.”

Understanding the Data

FuelConsumption.csv

I have downloaded a fuel consumption dataset, **FuelConsumption.csv**, which contains model-specific fuel consumption ratings and estimated carbon dioxide emissions for new light-duty vehicles for retail sale in Canada.

Link

- **MODELYEAR** e.g. 2014
- **MAKE** e.g. Acura
- **MODEL** e.g. ILX
  - 4WD/4X4 = Four-wheel drive
  - AWD = All-wheel drive
  - CNG = Compressed natural gas
  - FFV = Flexible-fuel vehicle
  - NGV = Natural gas vehicle
- **VEHICLE CLASS** e.g. SUV
- **ENGINE SIZE** e.g. 4.7
- **CYLINDERS** e.g 6
- **TRANSMISSION** e.g. A6
  - A = Automatic
  - AM = Automated manual
  - AS = Automatic with select shift
  - AV = Continuously variable
  - M = Manual
  - 3 – 10 = Number of gears

- **FUELTYPE**
  - X = Regular gasoline
  - Z = Premium gasoline
  - D = Diesel
  - E = Ethanol (E85)
  - N = Natural Gas
- **FUEL CONSUMPTION in CITY(L/100 km)** --> City and highway fuel consumption ratings are shown in litres per 100 kilometres (L/100 km) - combined rating (55% city, 45% hwy) is shown in L/100 km and in miles per imperial gallon (mpg)--> e.g. 9.9
- **FUEL CONSUMPTION in HWY (L/100 km)** e.g. 8.9
- **FUEL CONSUMPTION COMB (L/100 km)** e.g. 9.2
- **CO2 EMISSIONS (g/km)** --> Estimated tailpipe carbon dioxide emissions (in grams per kilometre) are based on fuel type and the combined fuel consumption rating --> e.g. 182 --> low --> 0

Goals of this analysis

There are two main goals for this analysis:

1. Performed Exploratory Data Analysis which will help me understand with what type of data I work on.
2. Performed Linear Regression Modeling together with Model Evaluation which will help me built a model to predict target - **CO2 EMISSIONS** variable with higher accuracy.

1. EDA:
    - 1.1. Data Exploration
    - 1.2. Log transforming skewed variables
    - 1.3. One Hot Encoding
    - 1.4. Data Visualization

2. Linear Regression Modeling:
    - 2.1. Linear Regression Model
    - 2.2. Lasso Regression
    - 2.3. Ridge Regression
    - 2.4. Model and Metrics Evaluation

Data Exploration

Using shape(), describe(), dtypes(), isnull() methods from pandas library I found out that in my dataset were 1067 rows and 13 columns.

5 of them was an object columns and the rest was numerical columns. I also learnt that there were no missing values in my dataframe so I didn't had to handle with this problem in this project.

Log transforming skewed variables

I defined a limit for skewed variables equal to 0.75. Above this I performed log transformation using **np.log1p**

Above this limit were 4 columns which are shown on the table at the right.

	Skew
FUELCONSUMPTION_HWY	1.263859
FUELCONSUMPTION_COMB	1.032592
FUELCONSUMPTION_CITY	0.900629
CYLINDERS	0.795754

One Hot Encoding

A significant challenge, particularly when dealing with data that have many columns, is ensuring each column gets encoded correctly.

Before I started to deal with one hot encoding I checked how many columns I will have if I will perform this function. It quickly turned out that I could create 739 extra columns which is a huge number.

So, I decided to take only two categorical columns **FUELTYPE** and **VEHICLECLASS** which seems for me be reasonable choice. I check again how many columns one hot encoding will produce and it ends with 18 extra columns.

At this stage of analysis I also decided to drop from my original data frame 4 columns: MODELYEAR, MAKE, MODEL, TRANSMISSION.

One Hot Encoding

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	CO2EMI
0	2.0	1.609438	2.388763	2.041220	2.251292	33	
1	2.4	1.609438	2.501436	2.163323	2.360854	29	
2	1.5	1.609438	1.945910	1.916923	1.931521	48	
3	3.5	1.945910	2.617396	2.312535	2.493205	25	
4	3.5	1.945910	2.572612	2.272126	2.451005	27	

5 rows × 27 columns

I finished this step with a new data frame named df_final which contains 1067 rows and 27 columns. This will be my data frame used for regression modeling.

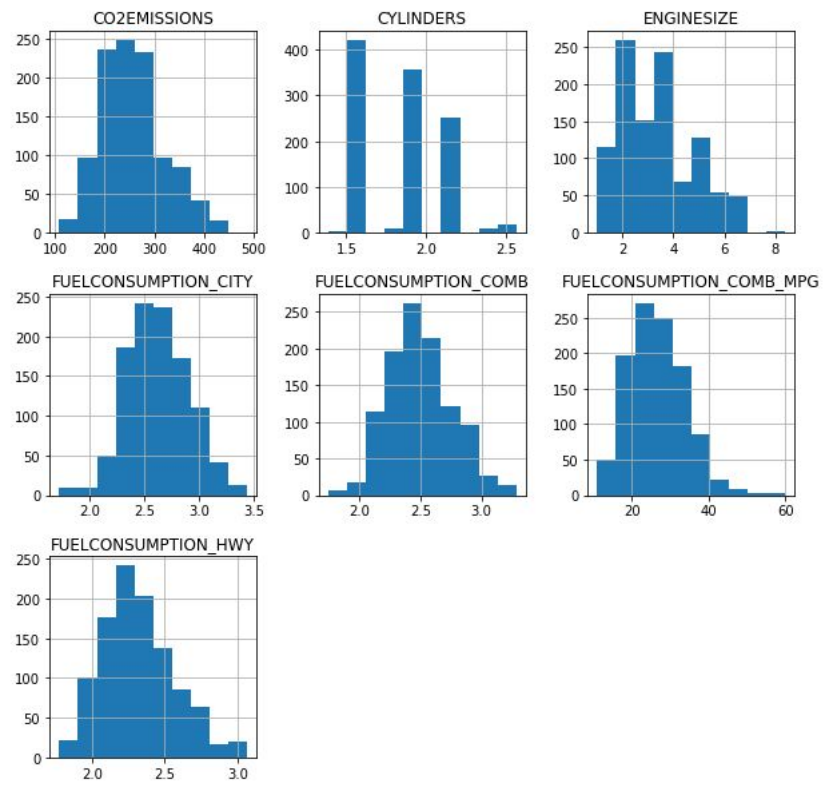
Data Visualization

In this subsection of my EDA part I used three different visualization methods to illustrate how my variables looks like.

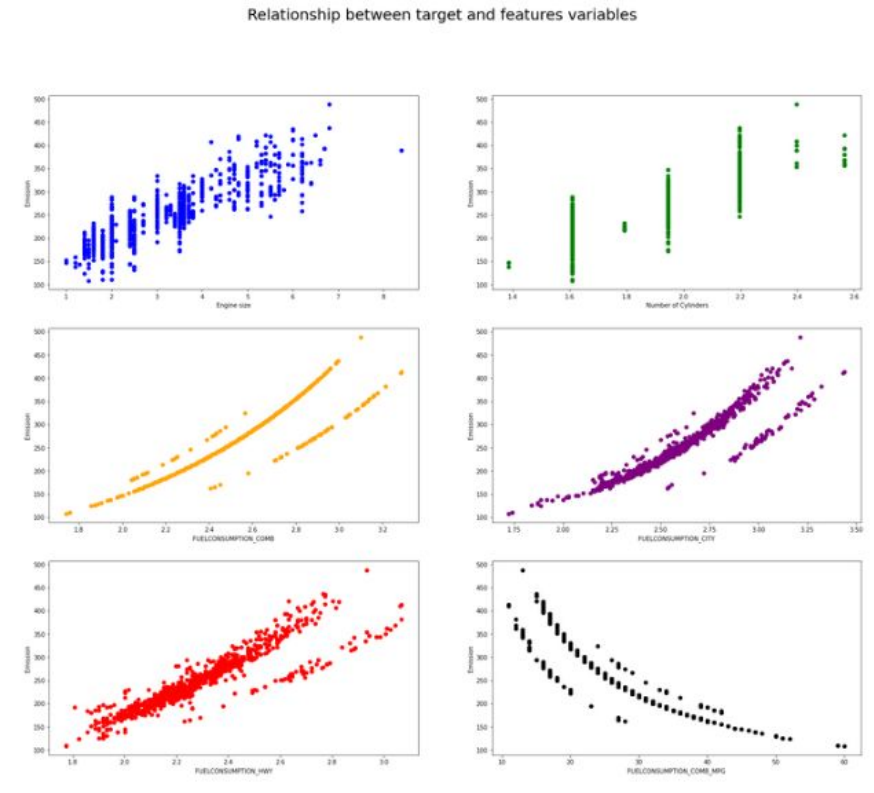
I used hist() method to show the distribution of chosen variables. Then plot this variables against the Emission, to see how linear their relationship is and finally I used seaborn function heatmap to illustrate correlation between variables which will be analysed in this project.

All results of this visualization will be shown in the next pages.

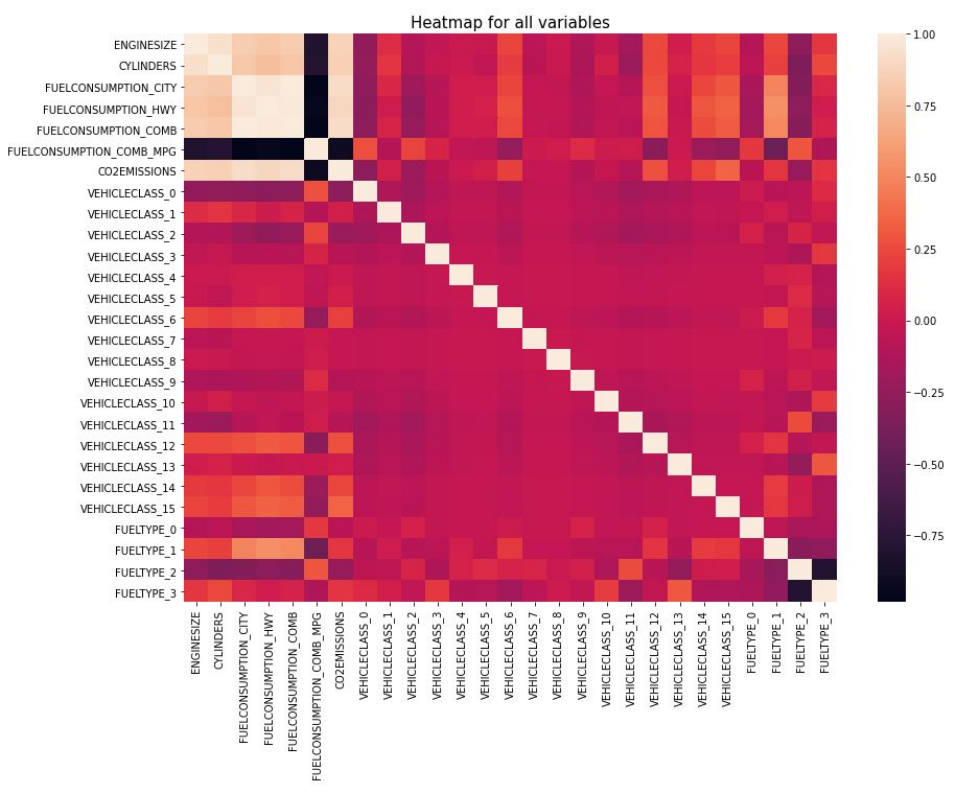
Data Visualization



Data Visualization



Data Visualization



Linear Regression Models

In this section I will present a final scores from three different models I implemented on my dataset and it was:

- Linear Regression Model
- Lasso Regression
- Ridge Regression

Linear Regression Model

First, I decided to use cross validation and by using KFold from sklearn.model_selection I split my data frame into 3 different train and test sets. Then, I implemented linear regression model for each of this sets and counted R-squared metrics for them.

I decided to check if scaling will bring me better results but as I expected for vanilla linear regression without regularization, scaling actually doesn't matter for performance.

Linear Regression Model

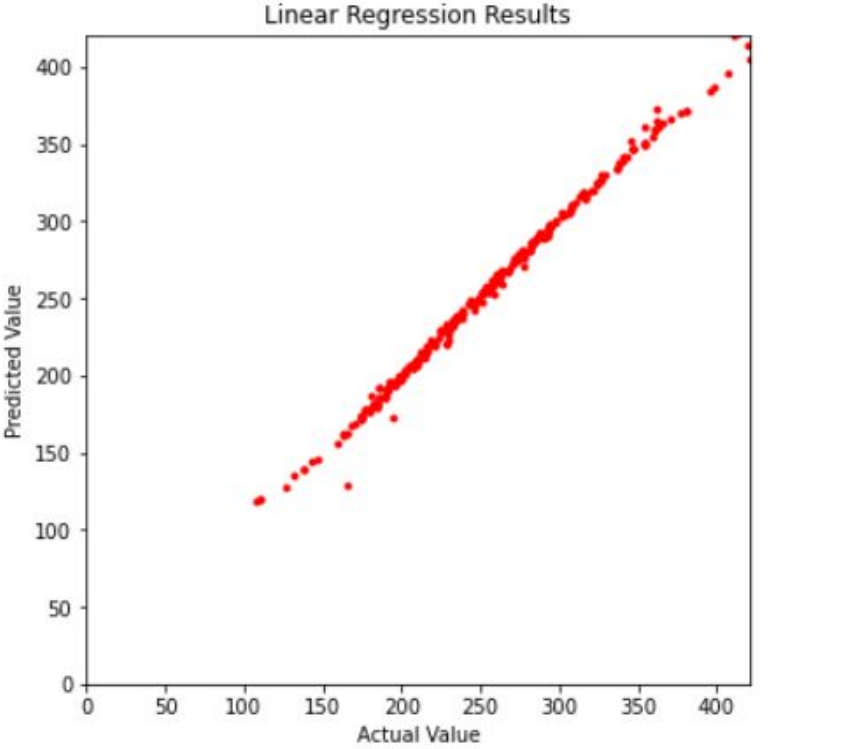
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=72018)
```

```
s = StandardScaler()
lr_s = LinearRegression()
X_train_s = s.fit_transform(X_train)
lr_s.fit(X_train_s, y_train)
X_test_s = s.transform(X_test)
y_pred_s = lr_s.predict(X_test_s)
linear_r2 = r2_score(y_pred_s, y_test)
```

Using tran_test_split method allowed me split dataset into 30% of test data and 70% of train data. As the code above present I used StandardScaler and LinearRegression object to find the R-squared metrics.

Next slide will show how well I was done this. I plotted y_test value against y_pred_s.

Linear Regression Model



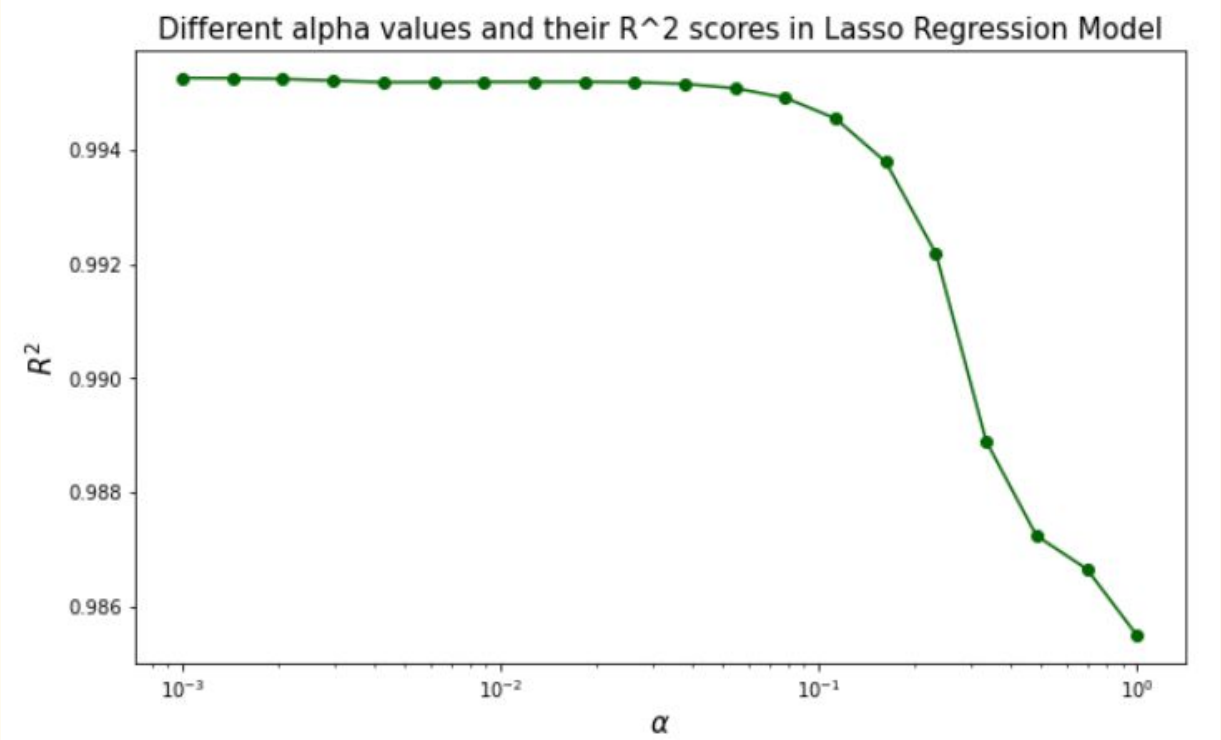
Lasso Regression

Hyperparameter tuning involves using cross validation (or train-test split) to determine which hyperparameters are most likely to generate a model that generalizes well outside of your sample.

I used here **PolynomialFeatures** object and different alphas values generated by np.geomspace to check which values will bring the highest score.

To improve the process I decided to use sklearn.pipeline function **Pipeline** where I passed 3 objects inside estimator: StandardScaler, PolynomialFeatures and selected models like Lasso and Ridge.

Lasso Regression



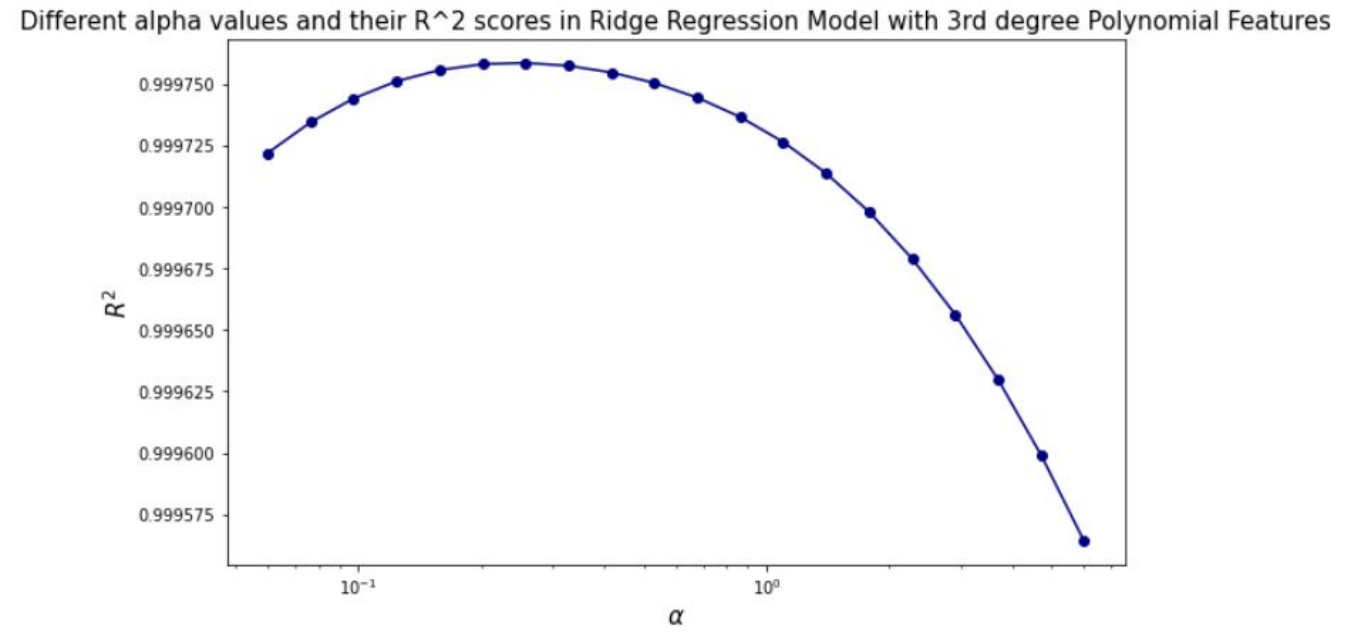
Lasso Regression

Another very useful features from sklearn.model_selection which helped me find the best parameters for my Lasso model was GridSearchCV.

It found clear that for Lasso I needed alpha equal to 0.06 and polynomial features degree equal to 3 to achieve the best results.

Ridge Regression

The same steps were performed for Ridge Regression. First, I wanted to know how different alpha values reflected to R-squared result. The results is represented by graph below.



Ridge Regression

Next by using Pipeline and estimator I created before for Lasso I performed GridSearchCV but this time passing Ridge as a model inside estimator.

The results I received using **grid.best_params_** method told me that in this case the best parameters were alpha equal to 0.26 and polynomial features degree equal to 3.

Key Findings - Metrics for Regression

In this section I will present findings and some basic metrics for each model I created earlier. I will concentrate on:

- **Mean Absolute Error (MAE)** represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.

- **Mean Squared Error (MSE)** represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

- **Root Mean Squared Error (RMSE)** is the square root of Mean Squared error. It measures the standard deviation of residuals.

- **R-squared (R2)** which is the proportion of variation in the outcome that is explained by the predictor variables. In multiple regression models, R2 corresponds to the squared correlation between the observed outcome values and the predicted values by the model. The Higher the R-squared, the better the model. The coefficient of determination or R-squared represents the proportion of the variance in the dependent variable which is explained by the linear regression model. It is a scale-free score i.e. irrespective of the values being small or large, the value of R square will be less than one.

Metrics for Regression

I created this table to better understand the results which each of models represent. As we can see each of models represent in the table achieve very high level of accuracy - above 99%. If we want to be very exact which the best model in this tally is, the Ridge Regression Model with R-squared metric equal to 99,99% win in this category.

	MAE	MSE	RMSE	R^2
Linear	2.541668	16.537494	4.066632	0.995469
Ridge	0.175911	0.055740	0.236093	0.999986
Lasso	0.549316	0.453746	0.673607	0.999887

Metrics for Regression

The remaining metrics: MAE, MSE and RMSE are those where the lowest value achieved by the model tell us that model perform the best.

Comparing to Lasso and Ridge it seems to be clear that plain Linear Regression model even with so high accuracy level performed the worst.

Both Lasso and Ridge achieve the same results with a slight difference in plus for Ridge Regression model. MAE, MSE and RMSE are almost double smaller than for Lasso.

Metrics for Regression

Now, the question which one to choose - Lasso or Ridge depends on business problem we try to solve.

Basically the main difference between those two models are the fact that Lasso is more likely to result in coefficients being set to zero. Because of that this model is better in terms of interpretations - it eliminate less significant variables (their coefficients set to 0).

On the other hand with Ridge model we gain on increasing the precision of our model.

Metrics for Regression

One of my goal of this analysis was to predict target variable - CO2 EMISSIONS and find the best accuracy model. In this project I am not concentrate so much on interpretations, rather the main concept how to implement machine learning algorithms (in this case Linear, Lasso and Ridge Regression) are more important for me. So, based on information I found I will choose Ridge Regression model as the best one in predicting CO2 EMISSIONS.

Model Evaluation

Model Evaluation

The models I created achieve high level of accuracy so it is hard to improve them more. But there are still some steps I can perform and play around with data I have. For example:

- ★ changing chosen variables and limit the number of it to let's say 6 instead 27 I worked with
- ★ as I only scaled variables in Linear regression I could add Polynomial Features to this model and check again for the model accuracy
- ★ hyperparameters tuning in Lasso and Ridge could be done by widening parameters used in GridSearchCV method (using different alphas, add more polynomial features degree or using more tuning models method)