



# Machine learning assignment

**Author: Teodor Miahil Moldoveanu**

|   |    |
|---|----|
| Introduction.....                         | 4  |
| Task 1.....                               | 4  |
| Finance consulting firm .....             | 4  |
| Scenario.....                             | 4  |
| Analysis .....                            | 4  |
| Healthcare organisation firm.....         | 5  |
| Scenario.....                             | 5  |
| Analysis .....                            | 5  |
| Social media platform scenario.....       | 5  |
| Scenario.....                             | 5  |
| Analysis .....                            | 5  |
| Autonomous delivery drone scenario.....   | 6  |
| Scenario.....                             | 6  |
| Analysis .....                            | 6  |
| Machine learning model selection.....     | 6  |
| Model evaluation .....                    | 6  |
| Model optimisation .....                  | 7  |
| Risk evaluation .....                     | 8  |
| Bias and Liability.....                   | 8  |
| Errors and Liability .....                | 8  |
| Suitability.....                          | 8  |
| Task 2.....                               | 9  |
| Data acquisition.....                     | 9  |
| Dataset cleaning.....                     | 11 |
| Data visualisation .....                  | 11 |
| Normalisation.....                        | 12 |
| Training data.....                        | 12 |
| Model performance evaluation methods..... | 12 |
| Mean squared error .....                  | 13 |
| Mean absolute error .....                 | 13 |
| Machine learning models.....              | 14 |
| Simple linear regression .....            | 14 |

|                               |    |
|-------------------------------|----|
| Model evaluation .....        | 17 |
| Polynomial regression .....   | 17 |
| Model evaluation .....        | 18 |
| Random forest regression..... | 19 |
| Decision trees .....          | 19 |
| Ensemble learning .....       | 19 |
| Bootstrapping.....            | 20 |
| Result .....                  | 20 |
| Model evaluation .....        | 20 |
| Optimisation .....            | 21 |
| Randomised search.....        | 21 |
| Conclusion.....               | 22 |
| References .....              | 23 |
| Appendix .....                | 24 |

# Introduction

Machine learning had become a prevalent component of our societies. It helps us make predictions and classifications based on existing data that have a measurable degree of accuracy. In this paper, analysis of machine learning models as well as the process in which machine learning models are selected, trained, and used are discussed in detail.

## Task 1

The identification and selection of machine learning models is done in accordance with the features that have to be implemented. To visualise and explain the machine learning model selection process, multiple scenarios are explored and explained below.

### Finance consulting firm

#### Scenario

As an employee of a financial consulting firm, I must develop a predictive machine learning model that can be used to predict stock prices, to help the firm buy and sell stock to maximise the firm's portfolio return.

#### Analysis

In this scenario the best types of models are supervised learning models. This is because the data regarding stocks and various financial indicators is labelled. Regression based machine learning models such as **Multiple linear regression**, **Polynomial linear regression**, and **Random Forest regression**, are the best fit for this kind of operations in which data is already classified. If data needs to be classified, unsupervised machine learning algorithms such as the clustering machine learning model algorithms **K-mean**, are the most suitable.

## Healthcare organisation firm

### Scenario

As a machine learning engineer within a healthcare organisation, I am assigned the task of creating a machine learning model that is segmenting the patients based on their symptoms. This must be done to increase the performance of the logistics as well as minimising the resource consumption.

### Analysis

In this scenario, because patients are part of different groups that are defined by their medical symptoms, classification-based machine learning models are the best fit to complete the task. The classification algorithms need to be unsupervised, because patients can have multiple symptoms and these symptoms can be unique and sometimes even un-accounted. For this scenario, unsupervised machine learning algorithms such as **K-mean** are the best fit for this specific task.

## Social media platform scenario

### Scenario

As a machine learning engineer within a social media platform, I am tasked with creating a machine learning model that is identifying and flags inappropriate content.

### Analysis

In this scenario, because user content is analysed, reinforced learning machine learning models are the best fit. This is because the content that users post is dynamic and very complex, and due to this, the machine learning model must classify the content on a policy-based system, which makes a reinforced learning machine model, the best fit for this scenario.

# Autonomous delivery drone scenario

## Scenario

As a machine learning engineer that is part of a futuristic logistic company, I am tasked with creating an autonomous drone delivery system. The drone system must be able to navigate complex urban environments to deliver the packages.

## Analysis

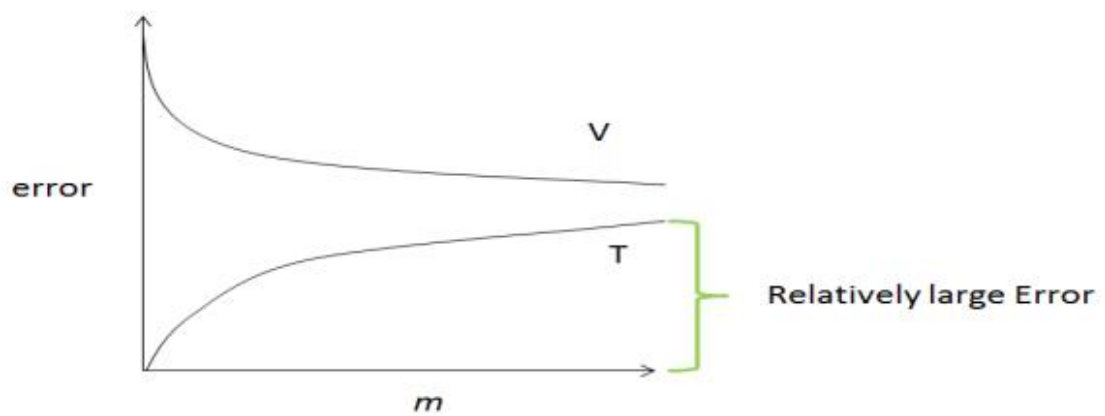
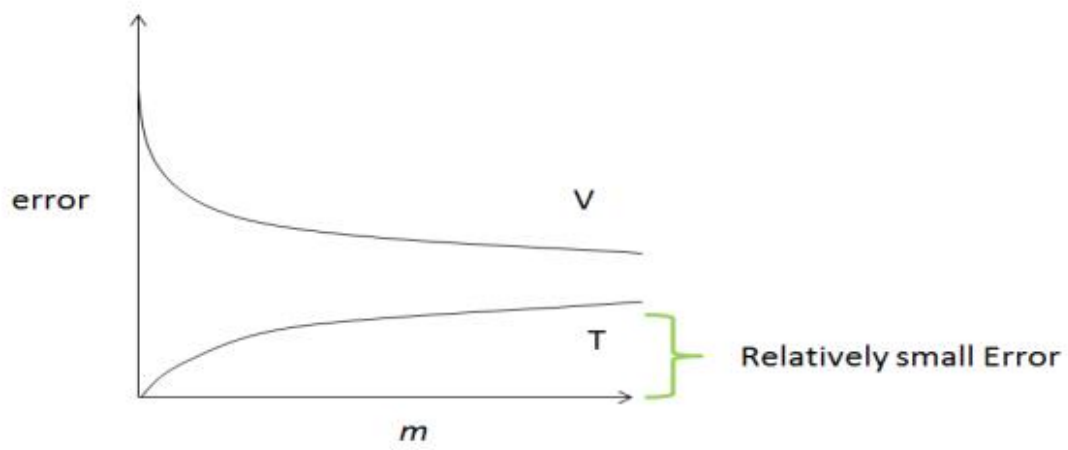
### *Machine learning model selection*

To conclude a precise analysis the closest self-driving machine learning systems that are in use, are implemented in self-driving cars. Self-driving cars use Deep Learning algorithms such as **Convolutional Neural Networks** that in turn use as sub-routines machine learning algorithms. Self-driving systems rely a lot on classification features to categorise objects and entities to take decisions. Because of the before mentioned fact, a classification machine learning model such as **K nearest neighbour** and **K mean** must be used to further enhance the capabilities of the system. The difference between the **K nearest neighbour** and **K mean** is the fact that **K nearest neighbour** is a supervised learning algorithm, whereas the **K mean** is an unsupervised machine learning algorithm. In the current scenario, because the data that this system is handling is dynamic both in visual terms and behavioural terms, the most suitable classification algorithm in this scenario is **K mean**.

### *Model evaluation*

The data used to train this machine learning system is composed out of both visual, and numerical/text data, and as a result, the data must be cleaned. After the data cleanup process is finalised, the data must be split, and the model's performance must be evaluated. The data split into an 80% portion for training and 20% portion for testing. This must be done to prevent the model overfitting. Model overfitting is a behavioural result of training the model on the whole dataset, without leaving any unknown variables, thus making the model unable to adapt to unforeseen scenarios. To evaluate the model, statistical error and precision formulas must be used to analyse and evaluate its performance. As an example, in a linear regression model, the loss of the machine learning model must be calculated using multiple statistical formulas, and these are:  $R^2$ , **mean squared error**, and **mean absolute error**.  $R^2$  is used to calculate the accuracy of the model, whereas both **mean squared error** and **mean absolute error** are used to calculate the error rate of the model from 0% to 100%.

### Model optimisation



**Figure 1 (Visualisation of the error rate)**

To minimise the loss of the model, hyperparameter tuning algorithms such as **random search** and **grid search** are the best way to minimise the error. These types of algorithms are sub-sampling the training and testing data in accordance with the number of hyperparameters set, and then train and test the model for each sub-sample train/test pair. Afterwards, they perform cross validation between the test results using loss calculation formulas such as  $R^2$ , **mean squared error**, and **mean absolute error**.

### *Risk evaluation*

#### *Bias and Liability*

The bias of the system can result in catastrophic and/or costly events. An example of a catastrophic event is the event in which a delivery drone would cause a traffic related and/or non-traffic related accident that results in the loss of human lives, which in turn is also a costly event, due to the lawsuits that can be launched against the company that produces the drones. This event could be caused by bias within the machine learning model, that can be in turn caused by model overfitting and a poor performance evaluation performed on the model. To minimise the bias, train/test data splitting as well as a thorough evaluation of the model's performance must be implemented accordingly with the technical specification of the model.

#### *Errors and Liability*

The same scenario as the one mentioned before can also be the cause of an error, which will lead in catastrophic and/or costly events. To minimise the risk of an error, model hyperparameter optimisation as well as a thorough evaluation of the model's performance must be implemented accordingly with the technical specification of the model.

#### *Suitability*

Not all machine learning models as well as, training data, and implemented features can fit all solution and/or scenarios. An example of a situation where the solution implemented is not suitable, is the event in which the delivery drones are trained on a dataset that is used to implement features related to self-driving car functionalities, whereas the drone will not need these features, thus resulting in a poor performance of the delivery drones, which can in turn lead to catastrophic and/or costly errors. A solution to this will be to test and evaluate the machine system and its features.

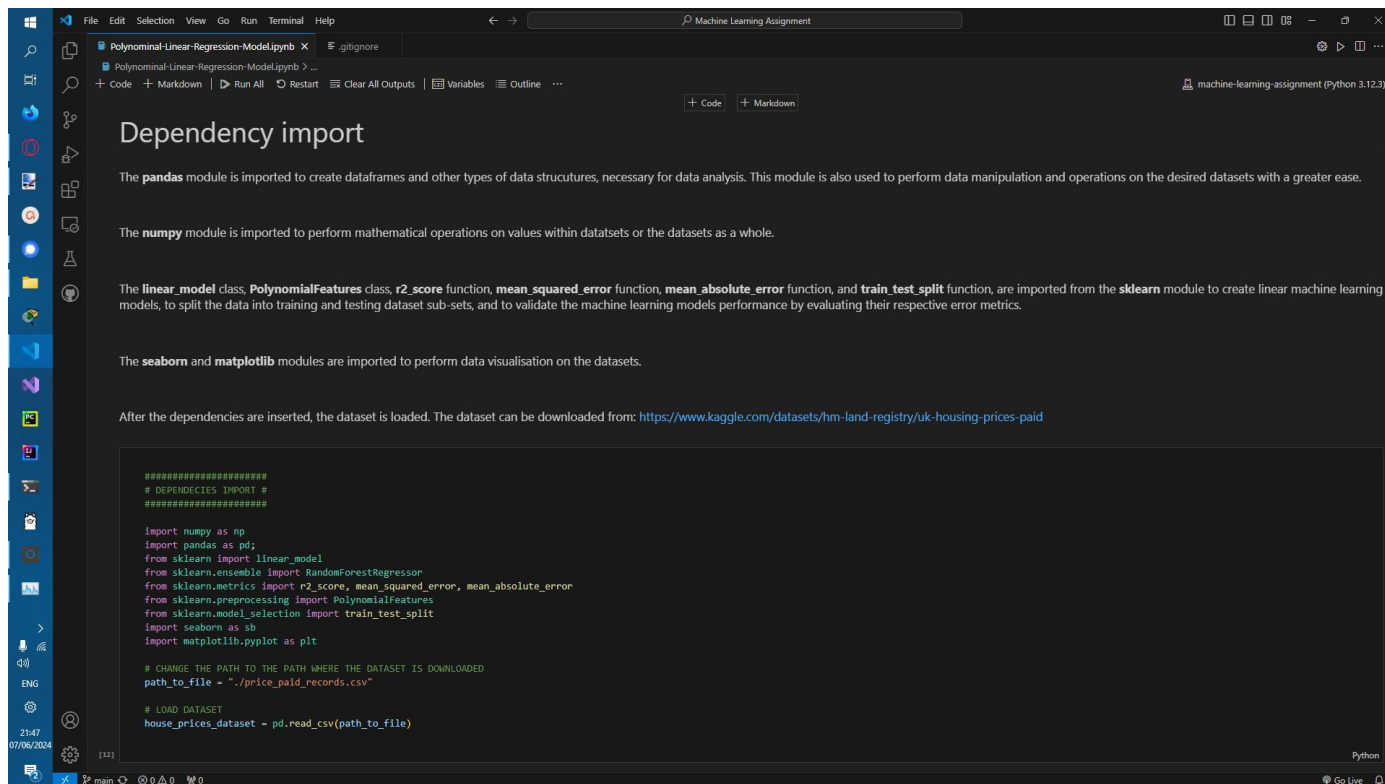


## Task 2

To create a regression-based machine learning model that can predict house prices, a dataset must be collected, data must be cleaned, data must be analysed, machine learning models must be trained on the data, and the models must be evaluated.

### Data acquisition

The data acquisition process was finalised by searching trusted sources that provide data regarding the prices of properties sold in different time periods. The data was extracted from the data repository **Kaggle**. The supplier of this dataset is **HM Land Registry**, which is a governmental agency of the UK government that is responsible for managing properties. After the dataset was downloaded, the data is imported into Jupyter Notebook, along with the dependencies required to run the data analysis, data cleaning, and the initialisation of machine learning models.



**Figure 2 (Dependency import in Jupyter Notebook)**

Because the dataset downloaded is approximately 2GB in size, an optional dataset minimisation process was added, to reduce the size of the dataset.

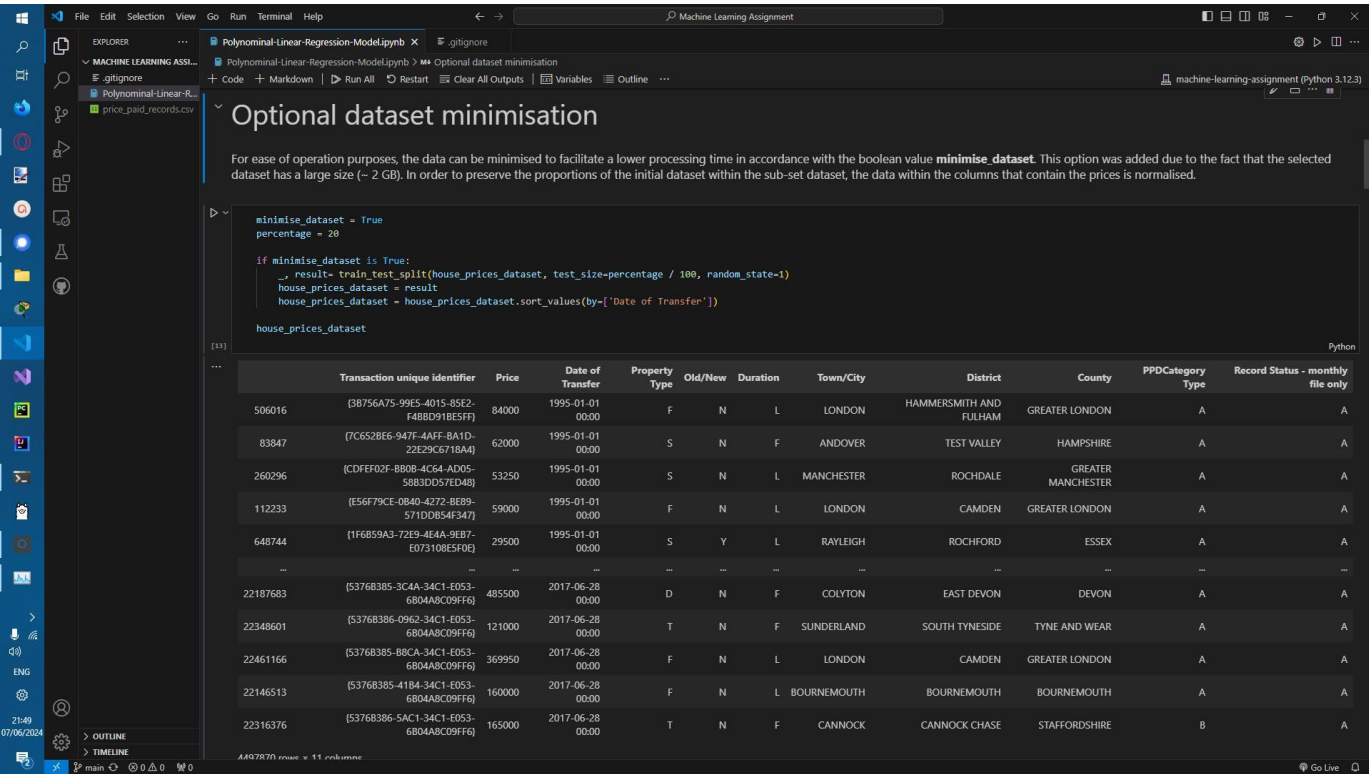
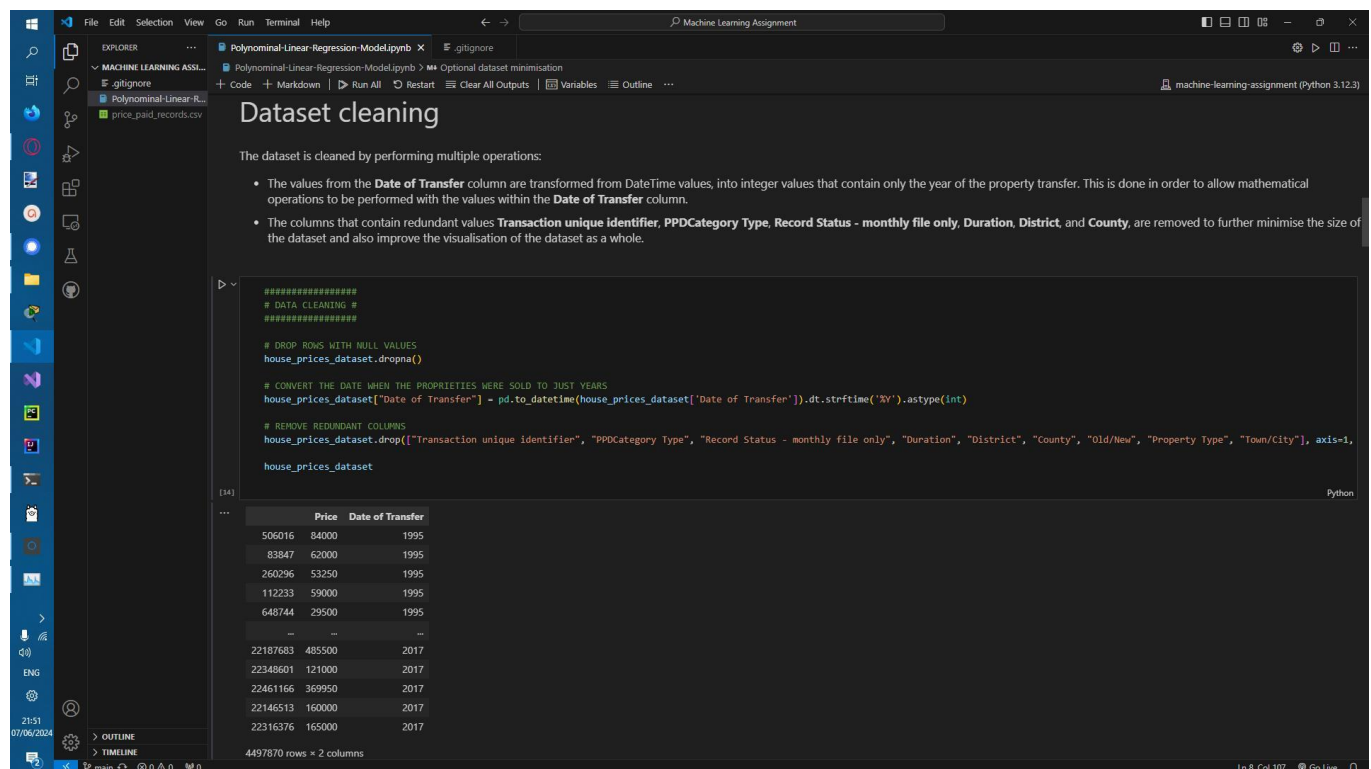


Figure 3 (Dataset minimisation in Jupyter Notebook)

## Dataset cleaning



**Figure 4 (Dataset cleaning in Jupyter Notebook)**

To initialise the dataset analysis, the dataset was cleaned of redundant columns and null values. The column that contains information regarding the date in which the properties were sold, was transformed from a date time format into a year only format. The before mentioned procedure was done to allow the machine learning models to calculate the evolution of the price against the evolution of time, by transforming the date time data into numerical values that can be used in mathematical operations.

## Data visualisation

The dataset was analysed to conclude the relationship between the prices of the properties and the price. A **lineplot** and **barplot** were used to analyse the relationship between the evolution of time and the prices of the properties, and the data visualisation is pointing to the fact that, prices are increasing in a linear manner as time passes.

## Normalisation

Because the numerical value difference between the prices and years is big, the dataset must be normalised. The normalisation procedure ensures that the difference between the values is minimal while maintaining their original meaning, thus minimising the bias that can be caused by this factor when machine learning models are processing the data (Galli, S. 2022).

$$x_{scaled} = \frac{x}{\max(x)}$$

Figure 5 (Galli, S. (2022))

## Training data

The data is split into two sub-sets, training data and testing data. The training data contains 80% of the original dataset, while the testing data contains 20% of the original dataset. This was done to prevent model overfitting. Model overfitting is an error caused by the fact that the machine learning model was trained with all available data, thus making it unable to adapt to new and unforeseen data, thus making the model useful only when the data from the dataset is used to make a prediction.

## Model performance evaluation methods

$$R^2 = 1 - \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^n (Y_i - \bar{Y}_i)^2}$$

Figure 6 (Newcastle University (2024))

$R^2$  is a statistical algorithm which is used to determine the accuracy of machine learning models (**Newcastle University (2024)**). It is calculated by subtracting the result of the division of the **sum of squares** and **total sum of squares** from 1, resulting in the formula

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - y_{Pred_i})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

The **sum of squares** is calculated by summing the squared result of the subtraction of the  $y$  value associated with the  $x$  value at the current index by the predicted  $y$  value associated with the  $x$  value at the current index,  $n$  times, resulting in the formula  $\sum_{i=1}^n (y_i - y_{Pred_i})^2$  (**Valchanov, I. (2018)**).

The total **sum of squares** is calculated by summing the squared result of the subtraction of the  $y$  value associated with the  $x$  value at the current index by the sample mean of the  $y$  values noted  $\bar{y}$ , resulting in the formula  $\sum_{i=1}^n (y_i - \bar{y})^2$  (**Thiagarajan, G.**).

### Mean squared error

$$MSE = \frac{\sum (y_i - \hat{y}_i)^2}{n}$$

**Figure 7 (Frost, J. (2021))**

The **mean squared error** is a statistical formula used to test the error rate of the machine learning models from a scale from 0% to 100%. This calculation is done by dividing the **sum of squares** and dividing the result by the total number of element indexes **Frost, J. (2021)**.

### Mean absolute error

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

**Figure 8 (Singh, V. (2023))**

The **mean squared error** is another statistical formula used to test the error rate of the machine learning models from a scale from 0% to 100%. This calculation is done by multiplying the division of 1 by the total number of element indexes, by the summation

of the absolute value of the subtraction of  $y$  value associated with the  $x$  value at the current index by the predicted  $y$  value associated with the  $x$  value at the current index  $n$ , resulting in the formula  $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - y_{Pred_i}|$  (Singh, V. (2023)).

## Machine learning models

### Simple linear regression

Linear regression-based machine learning models are using linear regression to predict values, by taking an input and giving an output.

$$y = \beta_0 + \beta_1 x.$$

**Figure 9 (Simple linear regression formula)**

The base formula for linear regression is  $y = \beta_0 + \beta_1 x$ , where  $y$  is the predicted  $y$  value,  $\beta_0$  is the  $x$  axis intercept, and  $\beta_1$  is the function's slope.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n x_i (y_i - \bar{y})}{\sum_{i=1}^n x_i (x_i - \bar{x})}$$

**Figure 10 (Beta 1) (Simple Linear Regression (n.d), p. 225)**

The  $\beta_1$  variable represents the slope, and the slope is representing the function's inclination.  $\beta_1$  can be found by dividing the summation of  $x_i (y_i - \bar{y})$  by  $\sum_{i=1}^n x_i (x_i - \bar{x})$ , where  $i$  is the current index and  $\bar{x}$  is the sample mean of the values within the  $x$  axis, and  $\bar{y}$  is the sample mean of the values within the  $y$  axis (Simple Linear Regression (n.d), p. 225).

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad \left| \quad \begin{array}{l} \text{WHERE } i = \text{current index} \\ , n = \text{number of samples} \\ \text{and } \bar{x} \text{ is the sample mean} \end{array} \right.$$

**Figure 11 (Sample mean x)**

$$\bar{y} = \frac{\sum_{i=1}^n y_i}{n} \quad \left| \quad \begin{array}{l} \text{WHERE } i = \text{current index} \\ , n = \text{number of samples} \\ \text{and } \bar{y} \text{ is the sample mean} \end{array} \right.$$

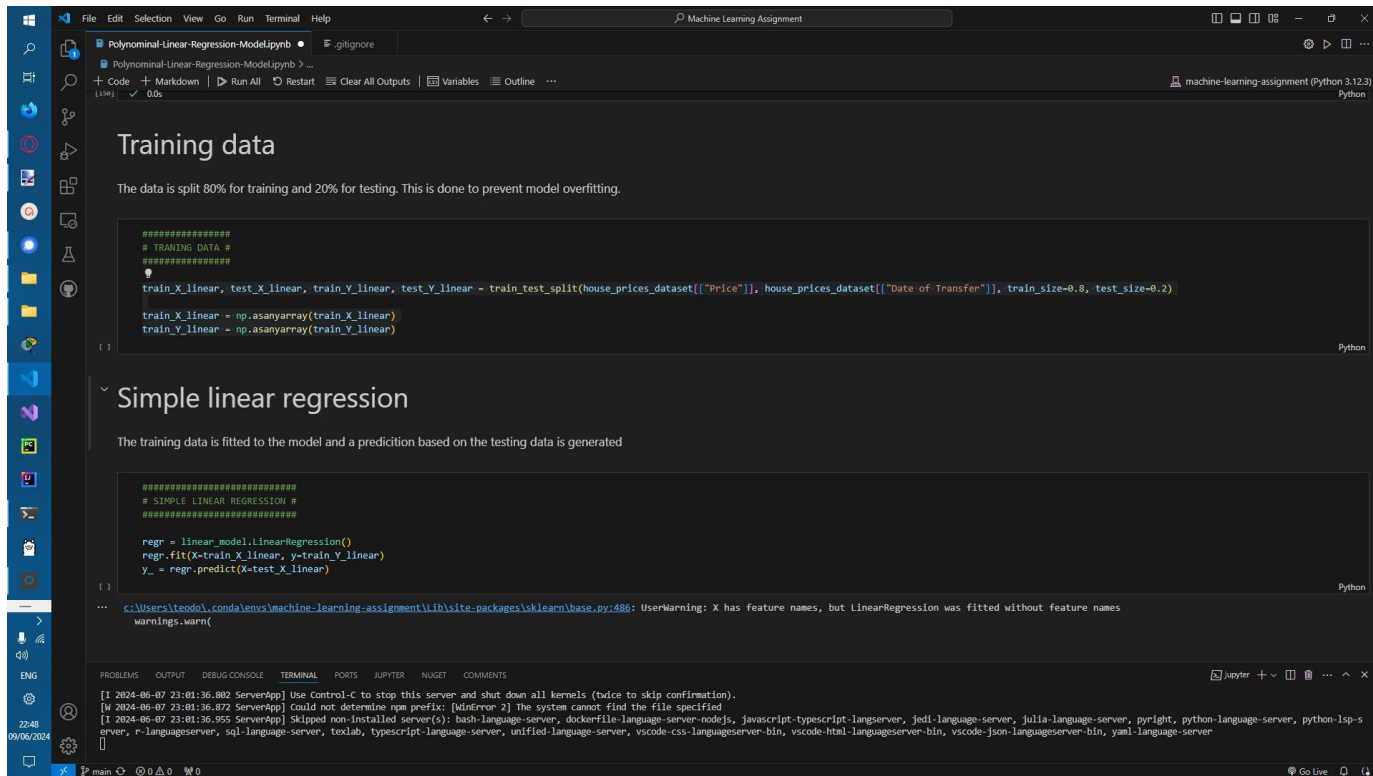
**Figure 12 (Sample mean y)**

The sample mean of both  $x$  and  $y$  can be obtained by summing all the terms within their own axis, respectively, and dividing by the total number of samples, respectively.

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

**Figure 13 (Beta 0) (Simple Linear Regression (n.d), p. 225)**

To obtain the  $\beta_0$  variable, the sample mean of the  $y$  axis denoted as  $\bar{y}$  must be subtracted by the product of  $\beta_1$  and the sample mean of  $x$ , denoted as  $\bar{x}$ , resulting in  $\beta_0 = \bar{y} - \beta_1 \bar{x}$  (**Simple Linear Regression (n.d), p. 225**). The  $\beta_0$  variable represents the function's intercept, and the intercept is the point of intersection of the function with the  $x$  axis.



**Figure 14 (Training data and simple linear regression implementation in Jupyter Notebook)**

To initialise the linear regression, the processed dataset is split into an 80% and 20% ratio, where 80% of the data will be used for training the models and 20% of the data will be used for testing the models. This split is done to prevent the overfitting of the models. After the data is split, the features are fitted into the model using the ***fit()*** method, the  $\beta_0$  and  $\beta_1$  variables are generated, and the predictions for each value within the subset of the dataset used for testing are generated.



```

Polynomial Linear Regression Model.ipynb
# SIMPLE LINEAR REGRESSION #
# TESTING THE ACCURACY #

mean_squared_error_res = mean_squared_error(y_pred-y_, y_true-test_y_linear)
mean_absolute_error_res = mean_absolute_error(y_pred-y_, y_true-test_y_linear)
r2_score_res = r2_score(y_pred-y_, y_true-test_y_linear)

print("r2: " + str(r2_score_res))
print("mean_absolute_error: " + str(mean_absolute_error_res))
print("mean_squared_error: " + str(mean_squared_error_res))

r2: 0.00040725196336627745
mean_absolute_error: 0.002567748861129185
mean_squared_error: 9.372521751795628e-06

```

**Figure 15 (Testing the accuracy in Jupyter Notebook)**

## Model evaluation

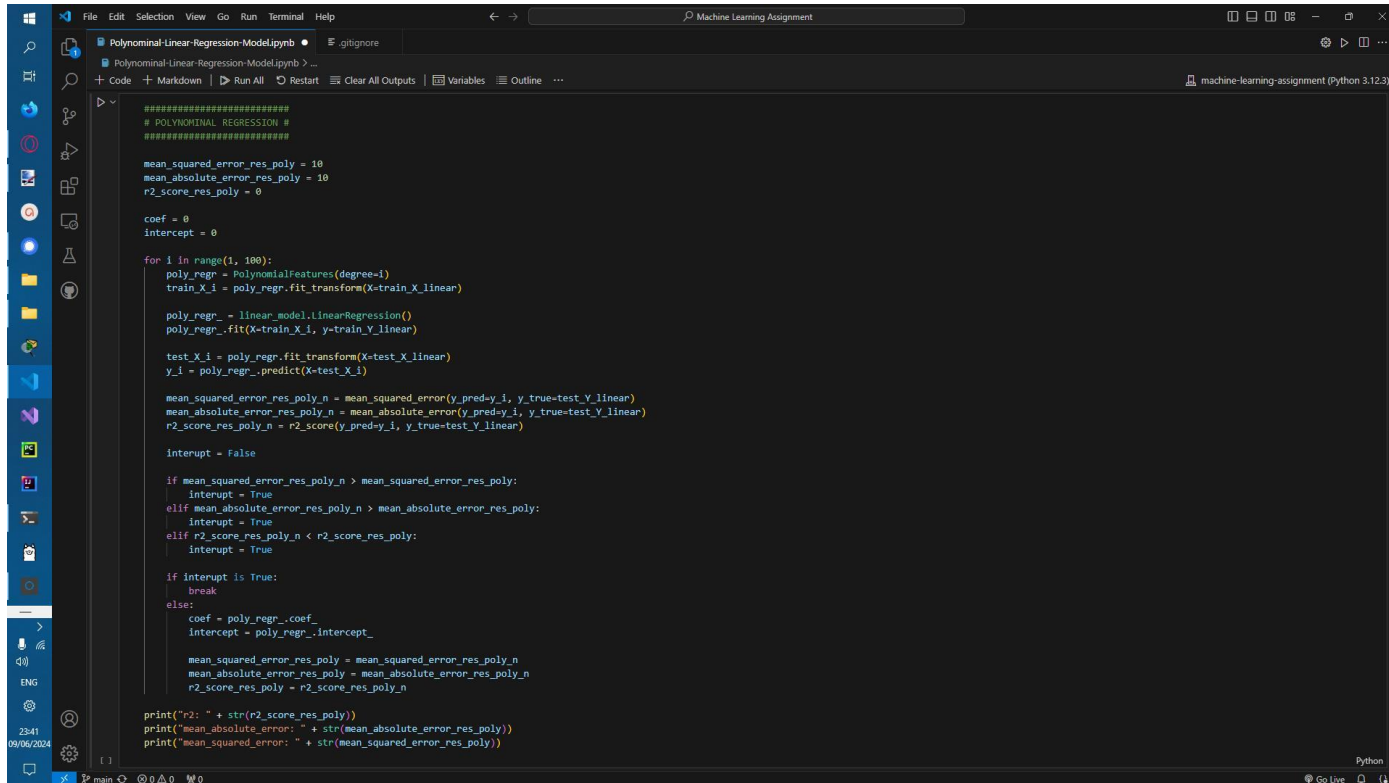
The model is evaluated using the **r2**, **mean absolute error**, and the **mean squared error** formulas. The **r2** formula is used to test the model’s accuracy, and both the **mean squared error** and **mean absolute error** are used to test the model’s error rate. The result shows that the linear regression model has an **r2** accuracy score of 0.04%, a **mean absolute error** score of 0.25%, and a **mean squared error** score of 0.0009%.

## Polynomial regression

$$\hat{y} = \beta_0 + \sum_{i=1}^n \beta_i^i \quad \left| \quad \begin{array}{l} \text{Where} \\ \beta_1^1 = \beta_1^1, \\ \beta_2^2 = \beta_1^2, \\ \beta_i^i = \beta_1^i \end{array} \right.$$

**Figure 16 (Polynomial regression formula)**

Polynomial regression is a more complex version of the linear regression. It is used to create more complex curves that can suit multiple scenarios where the data is not distributed linearly. The polynomial regression is calculated by adding the  $\beta_0$  value to a summation of  $\beta_i^i$  values, noted as,  $\sum_{i=1}^n \beta_i^i$  where  $i$  is the current index, and  $\beta_i^i$  is  $\beta_1^i$  raised to the power  $i$ , resulting in  $\beta_i^i$  **Jackson, D.S. (n.d.)**.



```

# POLYNOMIAL REGRESSION #
# POLYNOMIAL REGRESSION #

mean_squared_error_res_poly = 10
mean_absolute_error_res_poly = 10
r2_score_res_poly = 0

coef = 0
intercept = 0

for i in range(1, 100):
    poly_regr = PolynomialFeatures(degree=i)
    train_X_i = poly_regr.fit_transform(X=train_X_linear)

    poly_regr_ = linear_model.LinearRegression()
    poly_regr_.fit(X=train_X_i, y=train_Y_linear)

    test_X_i = poly_regr.fit_transform(X=test_X_linear)
    y_i = poly_regr_.predict(X=test_X_i)

    mean_squared_error_res_poly_n = mean_squared_error(y_pred=y_i, y_true=test_Y_linear)
    mean_absolute_error_res_poly_n = mean_absolute_error(y_pred=y_i, y_true=test_Y_linear)
    r2_score_res_poly_n = r2_score(y_pred=y_i, y_true=test_Y_linear)

    Interrupt = False

    if mean_squared_error_res_poly_n > mean_squared_error_res_poly:
        Interrupt = True
    elif mean_absolute_error_res_poly_n > mean_absolute_error_res_poly:
        Interrupt = True
    elif r2_score_res_poly_n < r2_score_res_poly:
        Interrupt = True

    if Interrupt is True:
        break
    else:
        coef = poly_regr_.coef_
        intercept = poly_regr_.intercept_

        mean_squared_error_res_poly = mean_squared_error_res_poly_n
        mean_absolute_error_res_poly = mean_absolute_error_res_poly_n
        r2_score_res_poly = r2_score_res_poly_n

print("r2: " + str(r2_score_res_poly))
print("mean_absolute_error: " + str(mean_absolute_error_res_poly))
print("mean_squared_error: " + str(mean_squared_error_res_poly))

```

**Figure 17 (Polynomial regression implementation Jupyter Notebook)**

To find the perfect curve, a for loop is initiated that runs from 1 to 100. Within this for loop at each iteration, the degree of the polynomial is assigned as the current index of the iteration, the model is trained and then tested. If the r2 value is smaller than the previous iteration, or the mean absolute error value is greater than the previous iteration, or the mean squared error is greater than the previous iteration, the loop is stopped, else the r2, mean absolute error, mean squared error, the  $\beta_0$  variable, and the  $\beta_1$  variable, are stored in their own assigned respective variables and the loop iterates further.

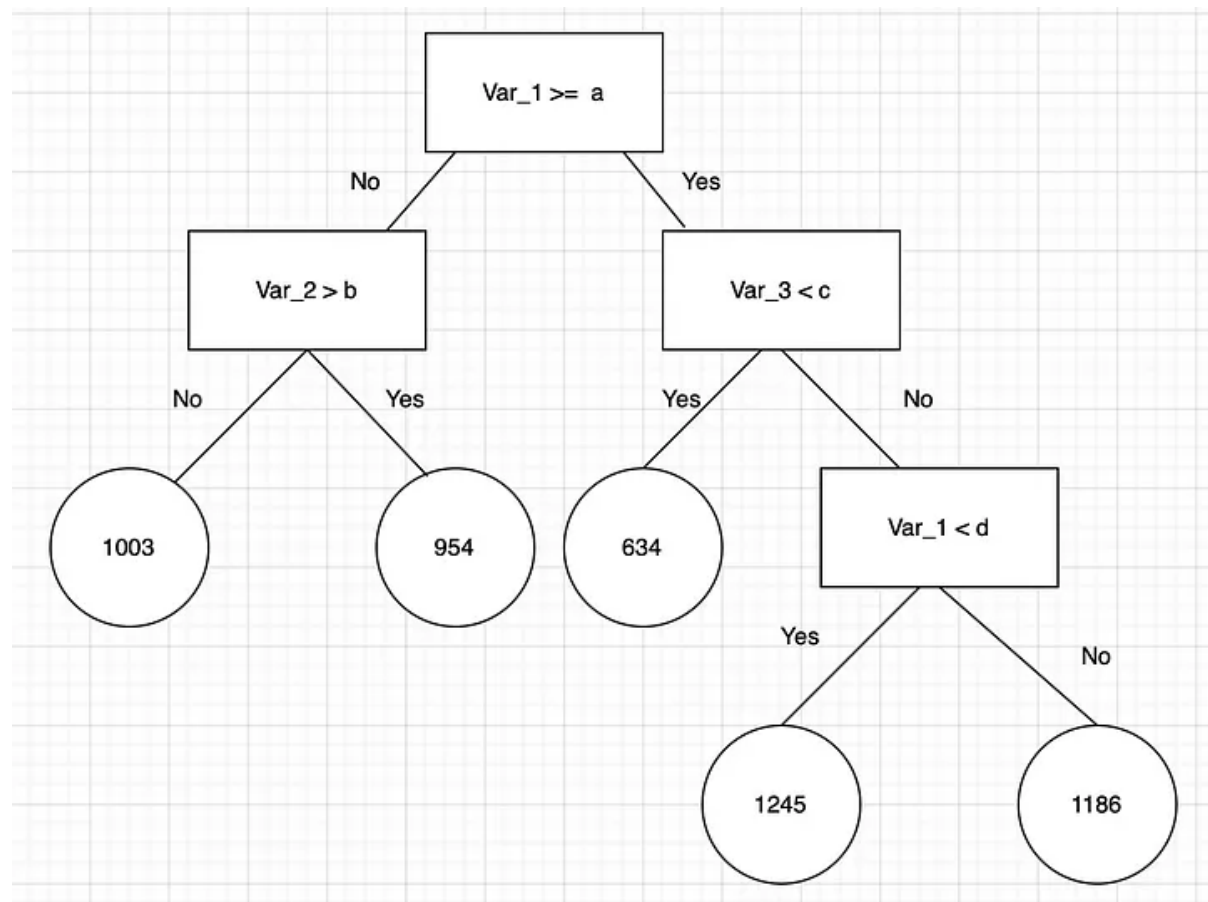
### Model evaluation

The model is evaluated using the **r2**, **mean absolute error**, and the **mean squared error** formulas. The result shows that the linear regression model has an **r2** accuracy score of 27%, a **mean absolute error** score of 0.21%, and a **mean squared error** score of 0.21%.

## Random forest regression

The random forest regression machine learning model is both a classification and a regression machine learning model that uses an aggregation of multiple machine learning methodologies and algorithms that are working together to achieve both classification-based and regression-based tasks (**Beheshti, N. (2022)**).

### Decision trees



**Figure 18 (Beheshti, N. (2022)).**

Decision trees are an algorithm that based on conditional statements verifying a value or multiple values, the algorithm is navigating branches and selecting different “leaves”, from top to bottom.

### Ensemble learning

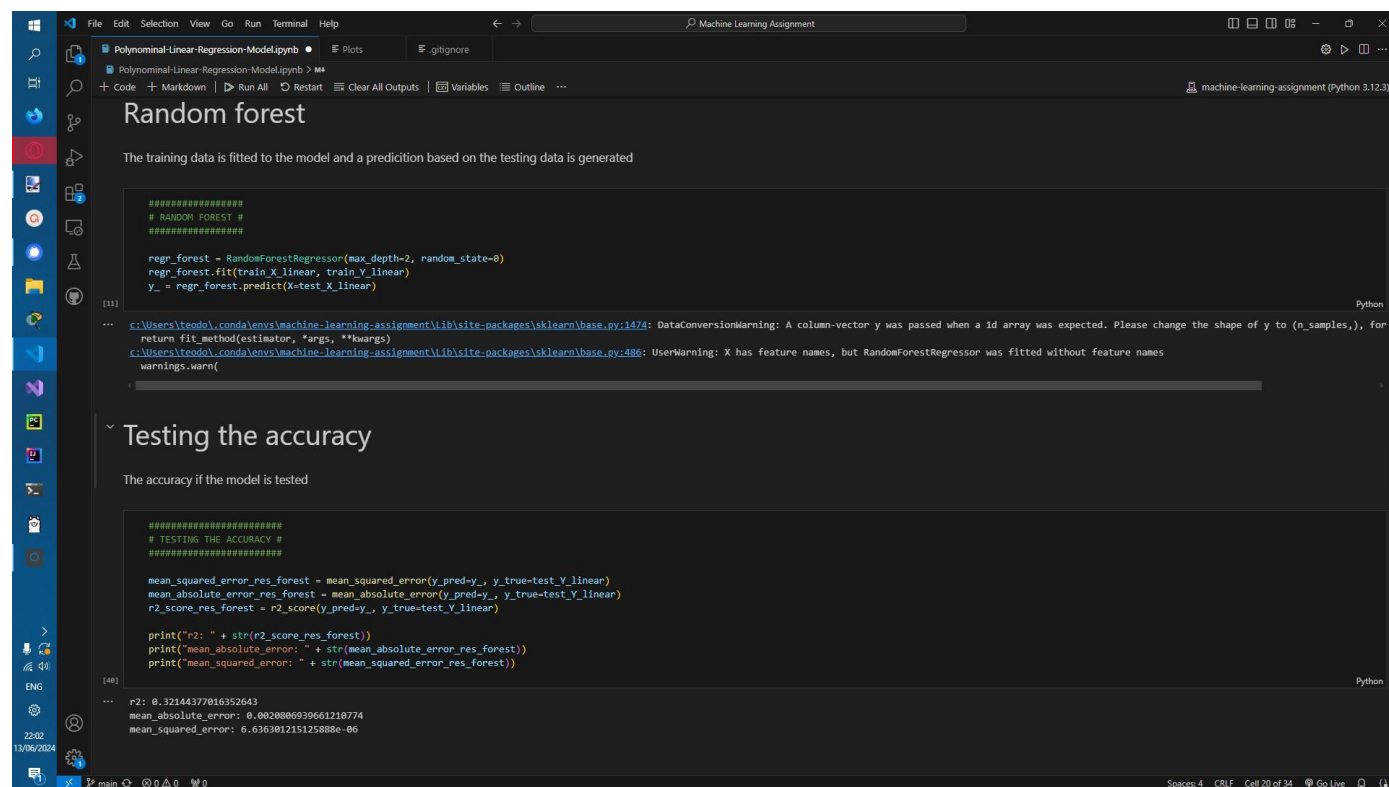
Ensemble learning is a machine technique in which multiple machine learning models are combined to complete a certain set of tasks by working as one singular machine learning model.

## Bootstrapping

Bootstrapping is a data sampling method in which data is sampled randomly from a dataset, and the resulting subsets are sampled recursively and randomly until the resulting subsets reach the desired size and/or after a certain number of iterations.

## Result

By combining the above-mentioned machine learning algorithms and techniques, the random forest regression algorithm is created and implemented. Within this implementation, the maximum depth of the decision tree is set to 2, and the random state value is set to 4, to allow to generate a predictable and reproducible output using a random number generator to sample the data.



```
Polynomial-Linear-Regression-Model.ipynb • Plots • .gitignore
+ Code + Markdown | ▶ Run All ⌂ Restart | Clear All Outputs | Variables | Outline ...
machine-learning-assignment (Python 3.12.3)

Random forest

The training data is fitted to the model and a prediction based on the testing data is generated

#####
# RANDOM FOREST #
#####

regr_forest = RandomForestRegressor(max_depth=2, random_state=0)
regr_forest.fit(train_X_linear, train_Y_linear)
y_ = regr_forest.predict(X-test_X_linear)

[31]

c:\Users\teodol\conda\envs\machine-learning-assignment\lib\site-packages\sklearn\base.py:1474: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
return fit_method(estimator, *args, **kwargs)
c:\Users\teodol\conda\envs\machine-learning-assignment\lib\site-packages\sklearn\base.py:486: UserWarning: X has feature names, but RandomForestRegressor was fitted without feature names
warnings.warn(

Testing the accuracy

The accuracy if the model is tested

#####
# TESTING THE ACCURACY #
#####

mean_squared_error_res_forest = mean_squared_error(y_pred-y_, y_true-test_Y_linear)
mean_absolute_error_res_forest = mean_absolute_error(y_pred-y_, y_true-test_Y_linear)
r2_score_res_forest = r2_score(y_pred-y_, y_true-test_Y_linear)

print("r2: " + str(r2_score_res_forest))
print("mean_absolute_error: " + str(mean_absolute_error_res_forest))
print("mean_squared_error: " + str(mean_squared_error_res_forest))

[48]

r2: 0.32144377016352643
mean_absolute_error: 0.0020806939661210774
mean_squared_error: 6.636301215125888e-06
```

**Figure 19 (Random Forest regression implementation Jupyter Notebook)**

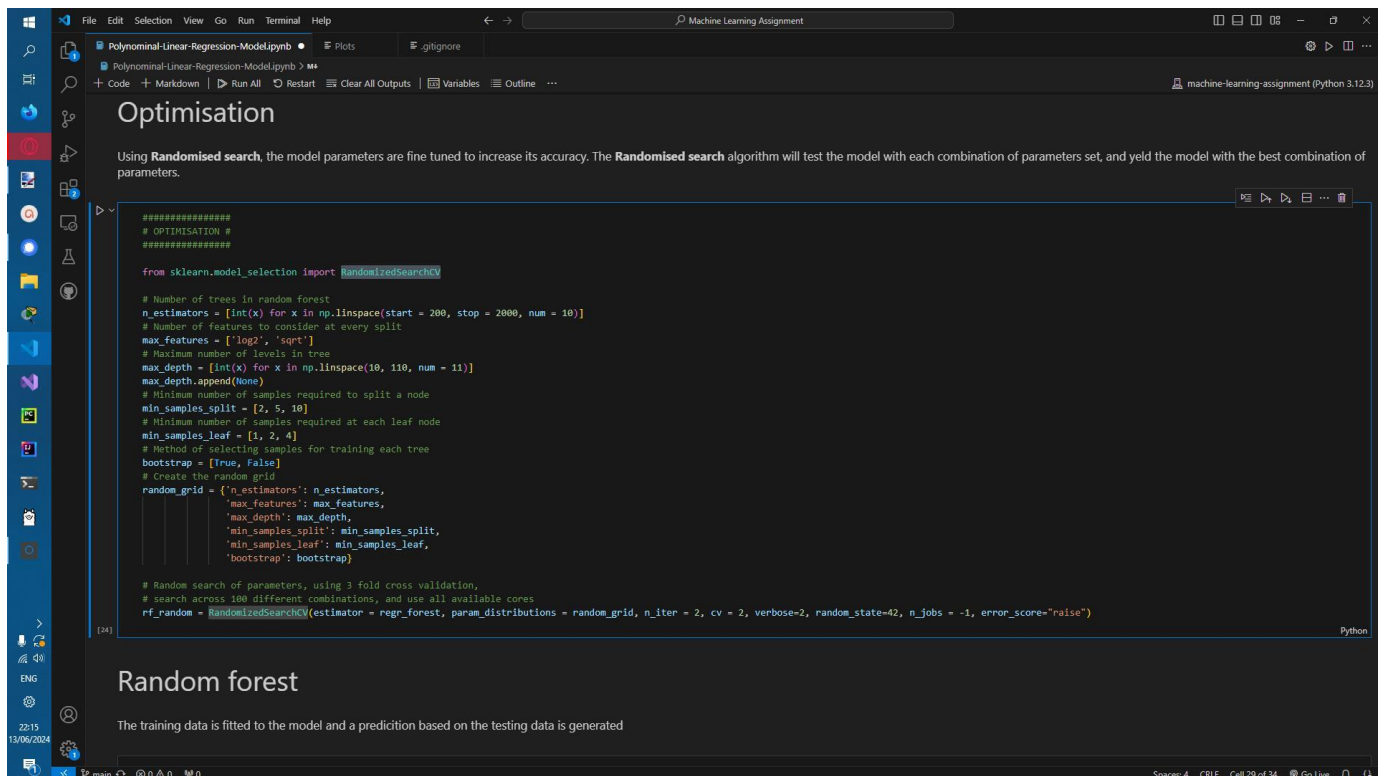
## Model evaluation

The model is evaluated using the **r2**, **mean absolute error**, and the **mean squared error** formulas. The result shows that the linear regression model has an **r2** accuracy score of 32%, a **mean absolute error** score of 0.20%, and a **mean squared error** score of 0.20%.

## Optimisation

### Randomised search

The randomised search algorithm is a model hyperparameter tuning algorithm that is used to tune the parameters of the machine learning models that it is optimising to make the model achieve its maximum level of accuracy by trying multiple combinations of hyperparameters and cross-validating them by splitting the data in multiple subsets, training the model on these subsets and evaluating the model using different loss calculation methods, such as **r2 (Koehrsen, W. (2018))**.



The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook is titled "Optimisation" and contains a Python code cell. The code implements a RandomizedSearchCV for a Random Forest model. It defines a parameter grid with various hyperparameters and uses 3-fold cross-validation to search for the best combination. The code is as follows:

```
#####  
# OPTIMISATION #  
#####  
  
from sklearn.model_selection import RandomizedSearchCV  
  
# Number of trees in random forest  
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]  
# Number of features to consider at every split  
max_features = ['log2', 'sqrt']  
# Maximum number of levels in tree  
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]  
max_depth.append(None)  
# Minimum number of samples required to split a node  
min_samples_split = [2, 5, 10]  
# Minimum number of samples required at each leaf node  
min_samples_leaf = [1, 2, 4]  
# Method of selecting samples for training each tree  
bootstrap = [True, False]  
# Create the random grid  
random_grid = {'n_estimators': n_estimators,  
               'max_features': max_features,  
               'max_depth': max_depth,  
               'min_samples_split': min_samples_split,  
               'min_samples_leaf': min_samples_leaf,  
               'bootstrap': bootstrap}  
  
# Random search of parameters, using 3 fold cross validation,  
# search across 100 different combinations, and use all available cores  
rf_random = RandomizedSearchCV(estimator = regr_forest, param_distributions = random_grid, n_iter = 2, cv = 2, verbose=2, random_state=42, n_jobs = -1, error_score="raise")
```

Below the code cell, the notebook displays the title "Random forest" and a message: "The training data is fitted to the model and a prediction based on the testing data is generated".

**Figure 20 (Random Search implementation Jupyter Notebook)**

```
Polynomial-Linear-Regression-Model.ipynb
+ Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['log2', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = regr_forest, param_distributions = random_grid, n_iter = 2, cv = 2, verbose=2, random_state=42, n_jobs = -1, error_score='raise')
```

### Random forest

The training data is fitted to the model and a prediction based on the testing data is generated

```
# Fit the random search model
rf_random.fit(train_X_linear, train_Y_linear)

y_ = rf_random.predict(X_test_X_linear)
```

Fitting 2 folds for each of 2 candidates, totalling 4 fits  
c:\Users\teodol\conda\envs\machine-learning-assignment\lib\site-packages\sklearn\base.py:1478: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for  
return fit\_method(estimator, \*args, \*\*kwargs)  
c:\Users\teodol\conda\envs\machine-learning-assignment\lib\site-packages\sklearn\base.py:496: UserWarning: X has feature names, but RandomForestRegressor was fitted without feature names  
warnings.warn(

**Figure 21 (Random Search implementation Jupyter Notebook)**

## Conclusion

The performance of the machine learning is visualised using a **barplot**. The result shows that the best performing model is the **Random Forest** model, and as a result this model will be chosen.

## References

Beheshti, N. (2022). *Random Forest Regression*. [online] Medium. Available at: <https://towardsdatascience.com/random-forest-regression-5f605132d19d>.

Frost, J. (2021). *Mean Squared Error (MSE)*. [online] Statistics By Jim. Available at: <https://statisticsbyjim.com/regression/mean-squared-error-mse/>  
[Accessed 14 Jun. 2024].

Galli, S. (2022). *Python Feature Engineering Cookbook*. [online] Available at: <https://learning.oreilly.com/library/view/-/9781789806311/45ab74c1-2d3d-4775-b1c7-a832233c528b.xhtml#b6c83a0a-3698-4dac-9cb8-611ebddd3419>  
[Accessed 14 Jun. 2024].

Jackson, D.S. (n.d.). *Chapter 7 Polynomial Regression | Machine Learning*. [online] *bookdown.org*. Available at: <https://bookdown.org/ssjackson300/Machine-Learning-Lecture-Notes/polynomial-regression.html>  
[Accessed 14 Jun. 2024].

Koehrsen, W. (2018). *Hyperparameter Tuning the Random Forest in Python*. [online] Medium. Available at: <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>  
[Accessed 14 Jun. 2024].

Newcastle University (2024). *Coefficient of Determination, R-squared*. [online] [www.ncl.ac.uk](http://www.ncl.ac.uk). Available at: <https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html>  
[Accessed 14 Jun. 2024].

Singh, V. (2023). *How to Calculate Mean Absolute Error - Shiksha Online*. [online] Shiksha.com. Available at: <https://www.shiksha.com/online-courses/articles/mean-absolute-error/> [Accessed 14 Jun. 2024].

Thiagarajan, G. (n.d.). *How to Calculate the Total Sum of Squares (SST)*. [online] Available at: [https://study.com/skill/learn/how-to-calculate-the-total-sum-of-squares-sst-explanation.html#:~:text=Total%20Sum%20of%20Squares%20\(SST\)%3A%20The%20SST%20is%20the,y%20i%20%E2%88%92%20y%20%C2%AF%20\)%202](https://study.com/skill/learn/how-to-calculate-the-total-sum-of-squares-sst-explanation.html#:~:text=Total%20Sum%20of%20Squares%20(SST)%3A%20The%20SST%20is%20the,y%20i%20%E2%88%92%20y%20%C2%AF%20)%202) [Accessed 14 Jun. 2024].

Valchanov, I. (2018). *Sum of Squares: SST, SSR, SSE*. [online] 365 Data Science. Available at: <https://365datascience.com/tutorials/statistics-tutorials/sum-squares/> [Accessed 14 Jun. 2024].

Simple Linear Regression. (n.d.). Available at: <https://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf> [Accessed 14 Jun. 2024].

## Appendix

**Jupyter Notebook source code:** <https://github.com/CSharpTeoMan911/Machine-Learning-Assignment>

**House Prices Dataset HM Land and Registry:** <https://www.kaggle.com/datasets/hm-land-registry/uk-housing-prices-paid/data>