LAB 8 – Decision Trees

In this lab session, we are going to implement decision trees with different parameters, visualize the results and try to examine the differences between them.

We are going to use sklearn's `DecisionTreeRegressor` class for this task. You can examine its documentation here.

Decision trees have many parameters which we can alter to control tree characteristics. We are going to investigate one of those in this lab: `max_depth`. Please browse through the documentation link given above to clearly understand what it refers to.

- (20 pts) Build the input and output:

Our .csv file is the same one we used for every lab except the previous one. The input columns are age, experience and power, while the output is the salary column. Build X and y the same way we did before (except this time, don't add a ones column to X because this is not a polynomial model). Furthermore, put the *headers* into a separate array.

<u>Divide your input and output into two parts: First 30 entries are going to be training data, the rest will become test data.</u>

- (30 pts) After building your X and y for train and test data, do the regression fitting:
    - Create a regression object (an instance of the `DecisionTreeRegressor` class. When creating the object, set the parameter `random_state` to 0, and `max_depth` to 1. Name this object `reg_1`.
    - Call `reg_1.fit(X_train, y_train)`.
    - Call `reg_1.predict(X_test)` and store the result. These are your predictions using this model.
    - Calculate and display MSE using the predictions and the initial salary values.
    - Print `reg_1.feature_importances_`.

- Call the following function:

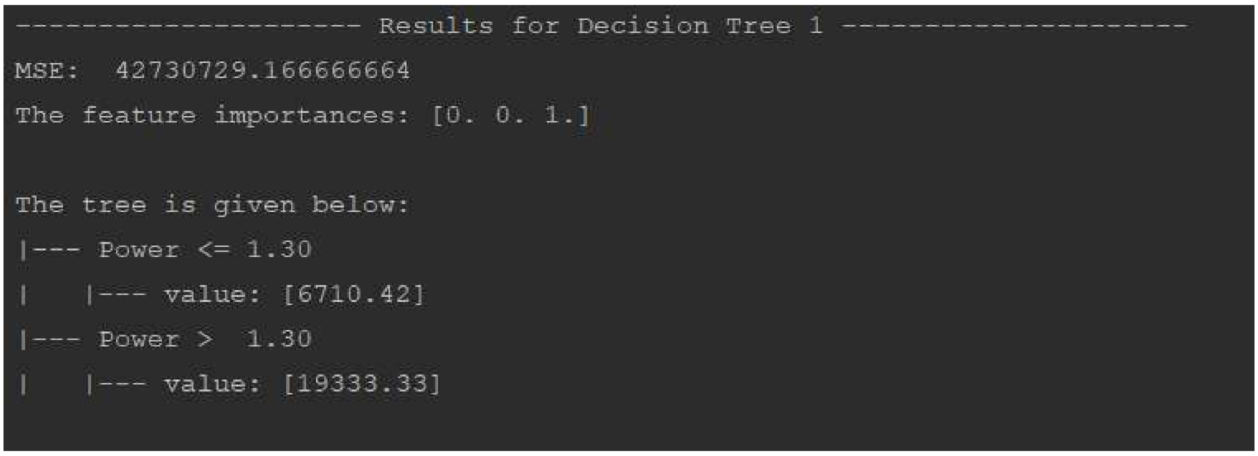
  ```
  export_text(reg_1, feature_name=titles)
  ```

  where `titles` is the array where you store the headers. Store and display the result. This function is part of the `sklearn.tree` library, therefore don't forget to import it.

- (10 pts) Repeat the entire process, except `max_depth` should be set to 3 when creating the regression object. Name this object `reg_2`.

- (10 pts) Repeat the entire process, except `max_depth` should be set to `None` when creating the regression object. This is the default value; therefore you can also remove the parameter entirely as an alternative. Name this object `reg_3`.

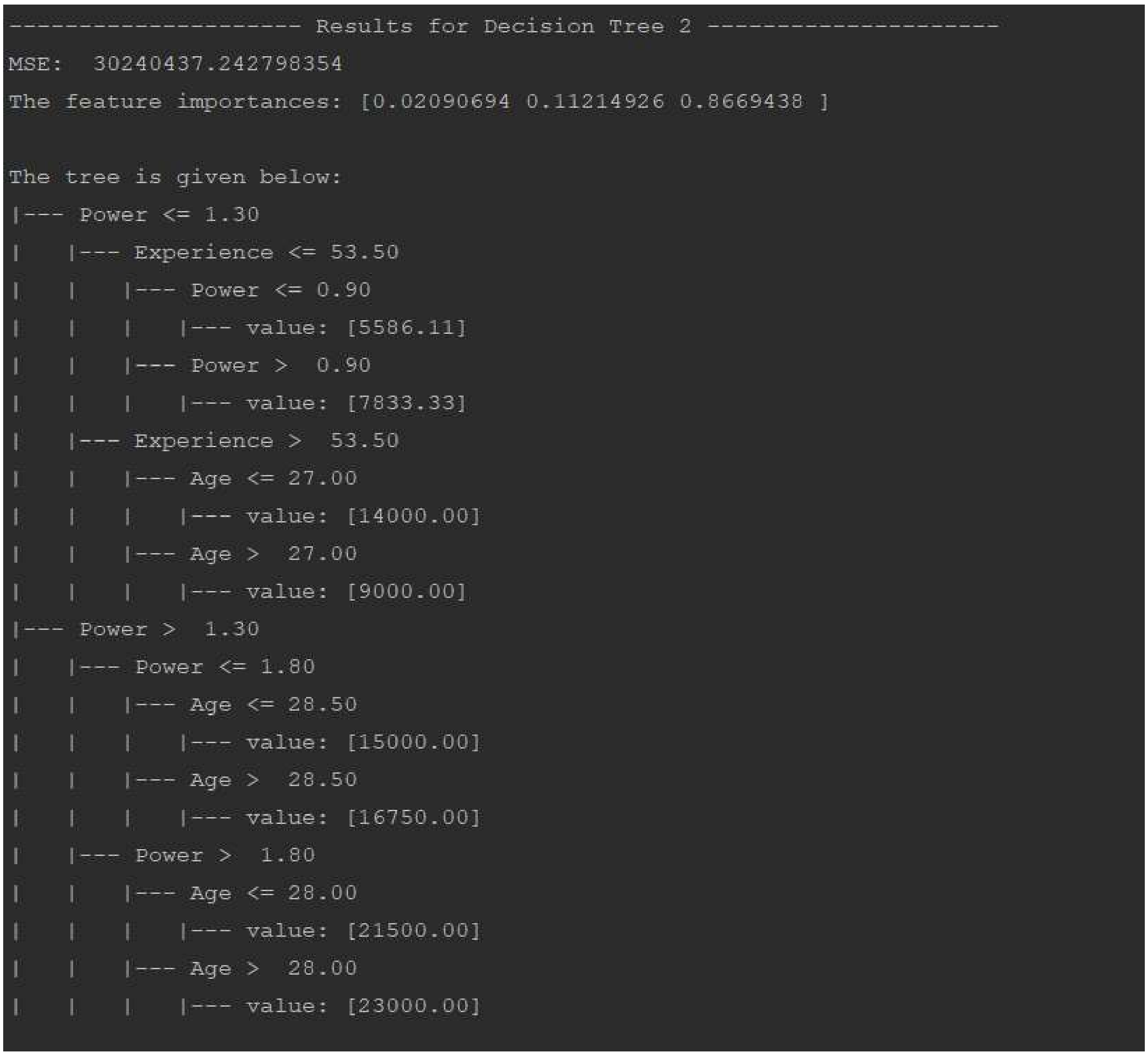
Here are the results for the first decision tree:



```
--------------------- Results for Decision Tree 1 ---------------------
MSE:  42730729.166666664
The feature importances: [0. 0. 1.]

The tree is given below:
|--- Power <= 1.30
|   |--- value: [6710.42]
|--- Power >  1.30
|   |--- value: [19333.33]
```

Here are the results for the second decision tree:



```
--------------------- Results for Decision Tree 2 ---------------------
MSE:  30240437.242798354
The feature importances: [0.02090694 0.11214926 0.8669438 ]

The tree is given below:
|--- Power <= 1.30
|   |--- Experience <= 53.50
|   |   |--- Power <= 0.90
|   |   |   |--- value: [5586.11]
|   |   |--- Power >  0.90
|   |   |   |--- value: [7833.33]
|   |--- Experience >  53.50
|   |   |--- Age <= 27.00
|   |   |   |--- value: [14000.00]
|   |   |--- Age >  27.00
|   |   |   |--- value: [9000.00]
|--- Power >  1.30
|   |--- Power <= 1.80
|   |   |--- Age <= 28.50
|   |   |   |--- value: [15000.00]
|   |   |--- Age >  28.50
|   |   |   |--- value: [16750.00]
|   |--- Power >  1.80
|   |   |--- Age <= 28.00
|   |   |   |--- value: [21500.00]
|   |   |--- Age >  28.00
|   |   |   |--- value: [23000.00]
```

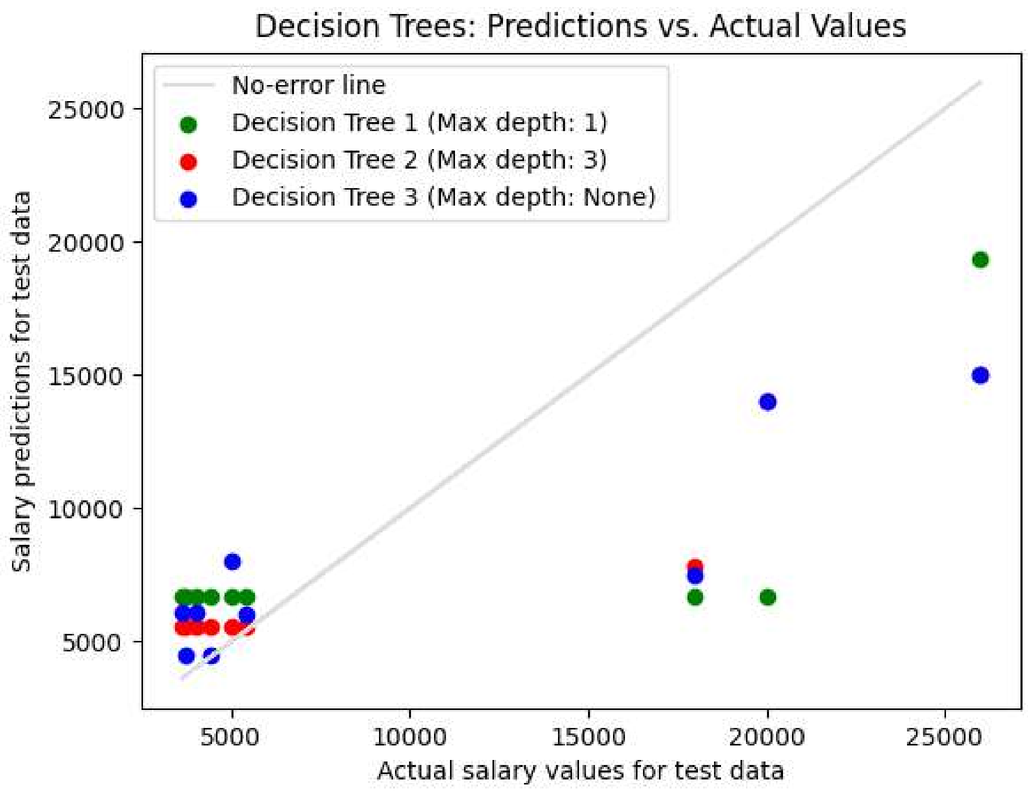
The results of the third decision tree are too long, and therefore not listed here.

- (30 pts) Now, the plotting:

  We will plot a “predictions vs actual values” graph. For each regression object, you should have a separate list of predictions. For each of these, plot them vs the actual salary values (`y_test`).

  Then, plot a line at $y = x$. This is the “no-error” line. We want the scatter plots to be as close to this line as possible.

Here is the resulting plot:



Continuing with the insight section:

Understanding how decision work is crucial for interpreting what we did in this lab. The console printouts show how the tree expands when the maximum depth is larger. The first tree only gives two possible outputs, since we only have one decision to make (and that decision is whether power is greater than 1.30 or not). As depth increases, we have more and more decisions to make, and the number of possible outputs we can generate grows larger.

Decision tree generation is a greedy algorithm, where we select the most important feature for a node first. You can see that in each tree, our first decision involved power, which is the most important feature (you can observe this in the “feature importances” array).

Even though in this example the last tree gave the best results, that doesn’t necessarily mean that it is a good model. The depth being too large leads to possible overfitting, and an optimal depth value can be found in most cases (cross-validation can be used to find such a depth value).

There are other tree parameters we can alter as well. For example, we can change the number of samples required to be in a leaf node. These parameters also change the decisions we make at each step, so feel free to experiment with those as well!

Random forests are usually better performing models compared to decision trees, since they generate *multiple* decision trees and output a combination of them. These trees are formed using bagging, which leads to a more reliable model in general.