# Programming Languages Project 1

Athina Bampzeli

March 2025

## 1 Question 1

**Choose one of the existing datasets and a) split your data into a train/test set, and b) select at least three classification or regression models (depending on the type of data) from the scikit-learn library (or other libraries) and apply them to the data. Try to tune the hyperparameters of the algorithm you use in order to demonstrate the phenomena of overfitting and underfitting. Do both phenomena appear? If not, why? Report the prediction accuracy on both the training and the test set, and justify the hyperparameter selection process.**

I chose to work with the dataset "CoalFiredPlantDataset". I separated the data to train and test sets as can be seen in Figure 1. As for the regression models, I tried LinearRegression, DecisionTreeRegressor and RandomForestRegressor. The code can be seen on my GitHub account.

```
#split data into train and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)

print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
print(f"Training set size: {X_train.shape[0]}, Test set size: {X_test.shape[0]}")
```

Figure 1: Train and test set

Overfitting occurs when a machine learning model learns the details and noise of the training data too well. As a result, it performs exceptionally well in the training data, but poorly in new unseen data. This happens when the model is too complex and the model memorizes the training data instead of learning general patterns. Underfitting occurs when a model is too simple and does not learn the underlying patterns in the training data. This leads to poor performance on both the training and test data because the model lacks the complexity needed to capture the relationships between variables. Overfitting and underfitting are generally considered opposite ends of the bias-variance tradeoff in machine learning, so they cannot occur at the same time for the same model on the same dataset. The Performance Metrics used are Mean Squared Error, $R^2$, Mean Absolute Error and Mean Absolute Performance Error.

The LinearRegression results suggest an almost perfect fit to both training and test data as the values of the MSE, MAE, MAPE are close to "0" and the values of the $R^2$ are close to "1" in both the train and test set. Performance metrics regarding the train and test sets can be seen in Table 1.

Table 1: LinearRegression Performance Metrics

| | MSE | $R^2$ | MAE | MAPE |
|---|---|---|---|---|
| Train set | 0 | 1 | 0.0005 | 0 |
| Test set | 0 | 0.9989 | 0.0017 | 0 |

The DecisionTreeRegressor results suggest that the model is overfitting to the training data. The model perfectly predicts the training data, as can be seen from the "0" values of the MSE, MAE and MAPE and the "1" value of the $R^2$. Although the error of the MSE, MAE and MAPE on the test set is small, the $R^2$ is -0.1572. A negative $R^2$ means the model performs worse than a simple mean prediction. Performance metrics before and after tuning the hyperparameters can be seen in Table 2. To reduce overfitting the solutions are to limit the max depth of the tree, increase the minimum samples per split and increase the minimum samples per leaf. The best hyperparameter values that were calculated via GridSearchCV can be seen in Table 3.

Table 2: DecisionTreeRegressor Performance Metrics

| | Train MSE | Train $R^2$ | Test MSE | Test $R^2$ |
|---|---|---|---|---|
| No tuning | 0 | 1 | 0.0039 | -0.1572 |
| Hyperparameter tuning | 0.0011 | 0.896 | 0.0027 | 0.2001 |

Table 3: DecisionTreeRegressor best Hyperparameters

| Max Depth | Min Samples Leaf | Min Samples Split |
|---|---|---|
| 5 | 4 | 2 |

The RandomForestRegressor results suggest that the model performs well on training data and that the test performance is decent but not ideal. The performance drop of the $R^2$ from training 0.9519 to test 0.6576 suggests some overfitting, but it is not extreme. Performance metrics can be seen in Table 4. To reduce overfitting the solutions are to try fewer estimators, increase the minimum samples per split and increase the minimum samples per leaf. The best hyperparameter values that were calculated via GridSearchCV can be seen in Table 5, although they result in an even smaller value of $R^2$ equal to 0.5727.

Table 4: RandomForestRegressor Performance Metrics

| | MSE | $R^2$ | MAE | MAPE |
|---|---|---|---|---|
| Train set | 0.0005 | 0.9519 | 0.016 | 0.0002 |
| Test set | 0.0012 | 0.6576 | 0.0275 | 0.0003 |

Table 5: RandomForestRegressor best Hyperparameters

| n estimators | Min Samples Leaf | Min Samples Split |
|---|---|---|
| 50 | 2 | 5 |

# 2 Question 2

**Choose a dataset with applications in chemical engineering that relates to your interests. Download the dataset and study it. Briefly describe the problem you have chosen to solve (data format, features, number of data points, number of classes, etc.). Can you load the data into Python? If not, describe the problem you encounter.**

The dataset that is chosen is the same used in Question 1, "CoalFiredPlantDataset". This dataset contains several features about a Coal Fired Power Plant for Thermal Performance Analysis and Prediction. The values of the features are numerical and are relative to inlet and outlet stream properties (temperature, pressure, flow rate, enthalpy), to flue gas properties, to boiler, turbine and preheater characteristics (efficiency, opacity, oxygen demand, leakage, power consumption). The dataset size is 5022 number of samples. The number of classes is 54. The dataset was imported successfully to the Colab environment and not Python using the command seen in Figure 2.

```
[ ]  from chelo import DatasetRegistry

     dataset = DatasetRegistry.get_dataset("CoalFiredPlantDataset")
     dataset.load_data()

     X, y = dataset.to_numpy()
```

Figure 2: Importing Dataset