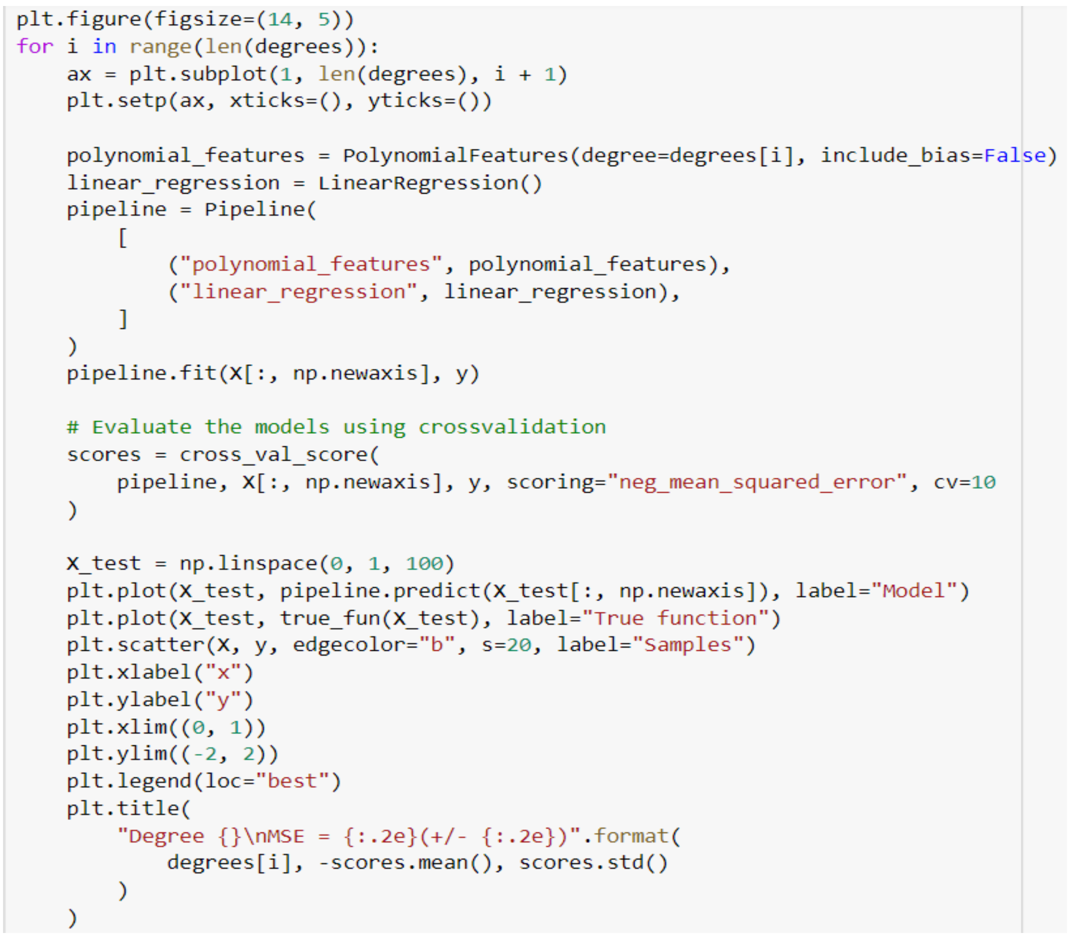
Name   : Mounika Ponnam
Student Id : 11610822

Understanding Overfitting and Underfitting

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
```

First of all we imported all required libraries,
numpy : It is used to perform various mathematical operations on arrays.
Matplotlib : It is a library to create interactive data visualizations.
Sklearn :It consists of efficient tools for statistical modeling and machine learning including regression, classification, clustering.



```
plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i + 1)
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline(
        [
            ("polynomial_features", polynomial_features),
            ("linear_regression", linear_regression),
        ]
    )
    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation
    scores = cross_val_score(
        pipeline, X[:, np.newaxis], y, scoring="neg_mean_squared_error", cv=10
    )

    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, edgecolor="b", s=20, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title(
        "Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(
            degrees[i], -scores.mean(), scores.std()
        )
    )
```

The code in the preceding paragraph explains the overfitting and underfitting issues as well as how to approximate nonlinear functions for linear regression using polynomial features. These models all share characteristics with polynomials to varying degrees. We already know that building up training samples requires more than a polynomial function of linear degree one. We refer to this as underfitting. Four-degree polynomial fits the situation perfectly. The model will overfit the training data, or learn the training data noise, if we utilize greater degrees. Cross-validation is used to quantitatively assess overfitting and underfitting.

1) According to you, why do overfitting and underfitting occur, and how do you resolve them? What is the difference between them?

When your model is too straightforward for your data, underfitting happens. When your model is too intricate for your data, overfitting takes place.

Overfitting : A model that overly accurately reproduces the training set of data is said to be overfit. When a model learns the information and noise in the training data to the point where it adversely affects the model's performance on fresh data, this is known as overfitting. This indicates that the machine learns concepts from the noise or random oscillations in the training data. These ideas don't apply to fresh data, which poses a difficulty for the models' capacity to generalize.

Underfitting : A model is said to be underfitting when it is unable to both model the training data and generalize to new data. A machine learning model that is underfit is not appropriate and will be visible since it will perform poorly on the training data.

Overcoming Overfitting : Different methods are frequently used by statisticians and ML experts to reduce overfitting in ML models. Cross-validation and regularization stand out as the most often used techniques. It has been demonstrated that these techniques work well for comprehending overfit data. Other methods exist in addition to these ones for removing overfitting from models.

Generalization: Make sure the data, for instance, leads to generalization rather than serving only as training data. More data can be fed into the model to achieve this. The accuracy of the model is also increased with more data. The model becomes computationally and memory-intensive as a result, though.

Data Augmentation: As a result, a different method known as data augmentation enters the picture. Instead of providing a ton of data, improvising and revising the already-present data can significantly reduce overfitting.

Overcoming Underfitting : Even though it is significantly less common in ML models, underfitting should still be taken into consideration. First off, there is a general lack of coherence between the data and model in this situation. This indicates that the model either performs very poorly with the training data or is far too simplistic to create a reliable learning pattern.
The solution, according to experts, is as simple as employing additional (excellent!) data for the project.

2)What kind of pattern did you analyze in the Train and Test score while running the code of overfitting?

Here, we can observe that while the test accuracy is noticeably low (66.99%), the train accuracy is quite high (99%), indicating overfitting. Before we employ data augmentation to solve overfitting, let's generate a few predictions.

3) What is cross-validation, and what did you analyze in a different type of validation that you performed?

Cross-validation is a statistical technique for assessing and contrasting learning algorithms that involves splitting the data into two sections: one for learning or training a model and the other for model validation.

K fold cross validation : One strategy to enhance the holdout method is K-fold cross validation. This approach ensures that our model's score is independent of how we chose the train and test sets. The holdout approach is applied k times while the data set is separated into k subsets. Let's proceed in the following order:

1. Divide your dataset into k number of folds at random (subsets)
2. Build your model using k - 1 folds of the dataset for each fold in your dataset. Then, test the model to see if it works for the kth fold.
3. Continue doing this until all k-folds have served as the test set.
4. The cross-validation accuracy, which is defined as the average of your k recorded accuracy, will be used as your model's performance metric.

This strategy typically yields a less biased model compared to other methods since it makes sure that every observation from the original dataset has the opportunity of appearing in the training and test set. If we have less input data, it is one of the finest strategies.

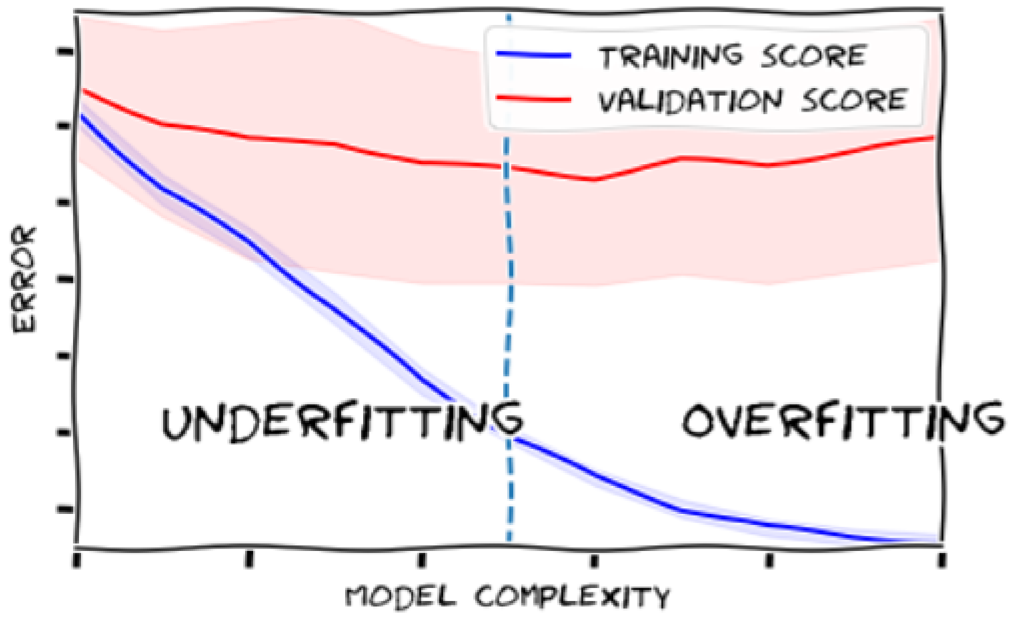
The drawback of this approach is that the evaluation process requires k times as much computing because the training algorithm must be run k times from the beginning.

Repeated k-fold cross-validation : It provides strategies for boosting the machine learning model's predicted performance. It determines the mean of all folds over all runs by repeatedly performing cross validation.

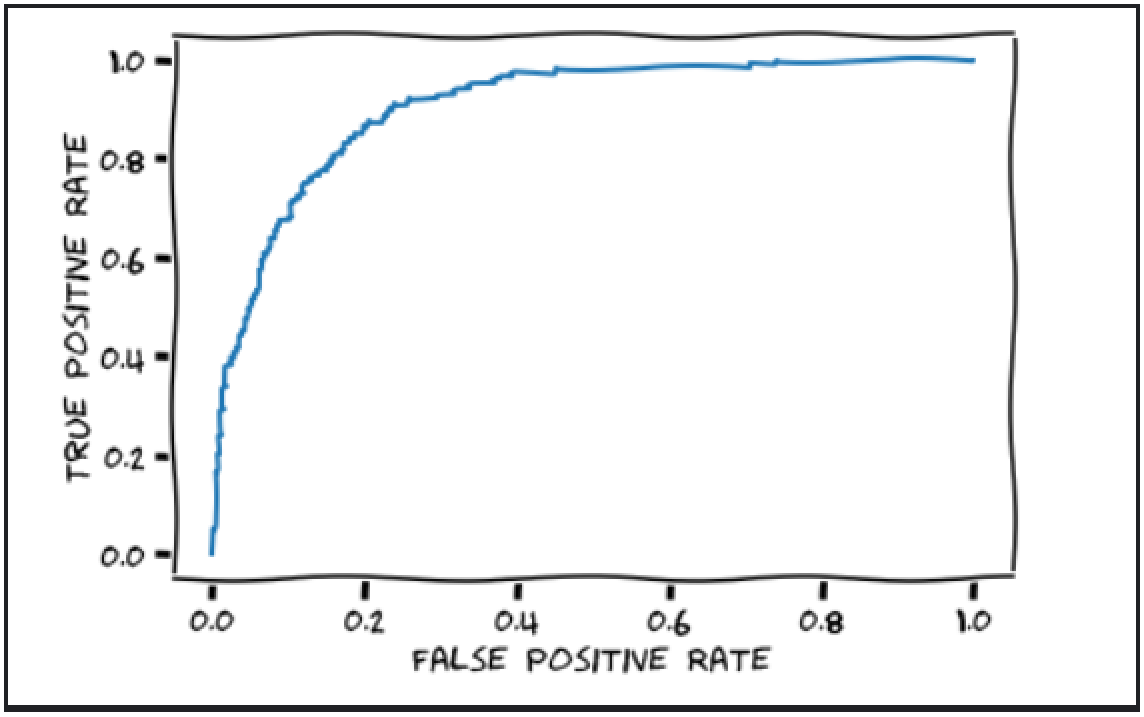
Leave-one-out cross validation : P value will be initially set to 1. Now that we have n number of choices for n data points and p = 1, the approach is significantly less exhaustive.

4) Explain the analysis from the generated ROC and validation curve and what they represent?

Validation curve: While using this curve as a backend cross-validation executes. The most interesting thing in the validation curve is that we do not need to divide the data into training and testing dataset,because it will be automatically done inside the validation curve. It will happen based on the fold number in cv. If without cross validation, we use a splitting function, the accuracy score will automatically vary depending on the random state.



<u>ROC curve</u> : (Receiver Operating Characteristic) It is a graph which displays the classification model performance at all thresholds of classification.ROC graph consists of two parameters. True Positive Rate and False Positive Rate.


TRUE POSITIVE RATE
FALSE POSITIVE RATE