Report

Machine Data & Learning Assignment-1

Saransh Rajput : 2018114016
Shivam Mangale : 2018101008

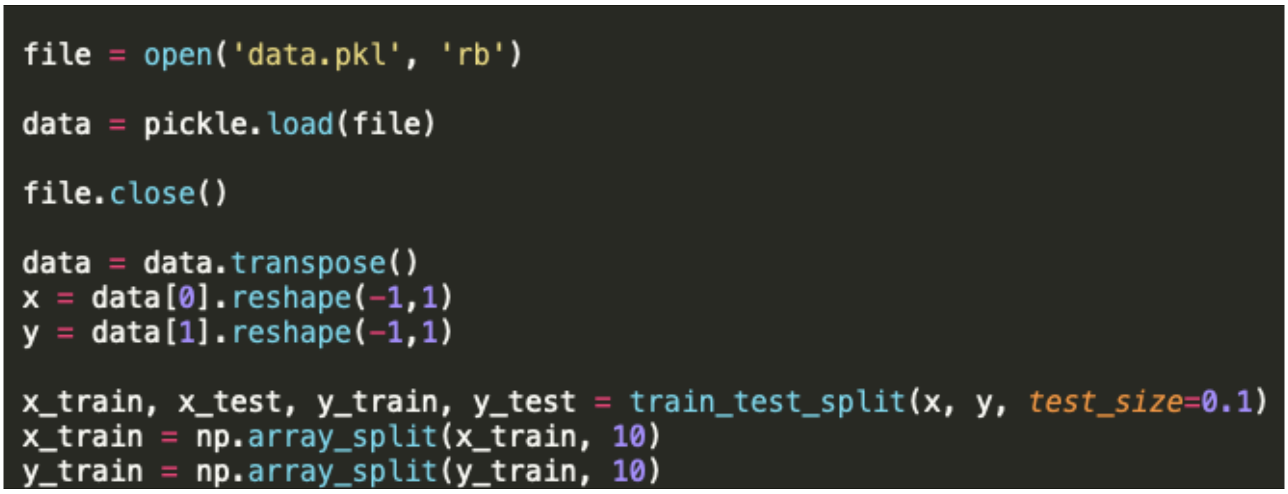
Introduction

Bias term measures how well our choice of a model, for example linear models, can approximate the ground truth in a best-case scenario.
Variance term measures how tightly clustered trained models are around this best-case scenario. Low variance means that all trained models are very close to the best-case scenario and high variance means that they are spread out.
There is a trade-off involved and the goal generally is to try to minimise Variance and Bias within the given constraints.

Question 1

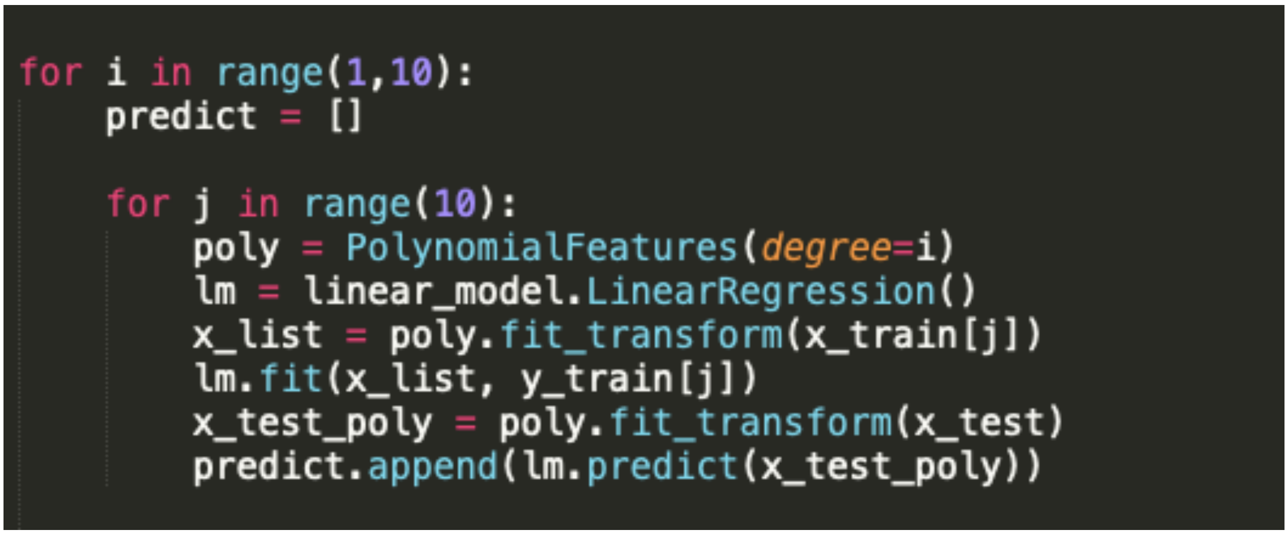
Task: Using sklearn's linear_model.LinearRegression().fit() for finding the appropriate coefficients with the default parameters for the given data.

The given file is loaded as pickle and is split into test data and training data.



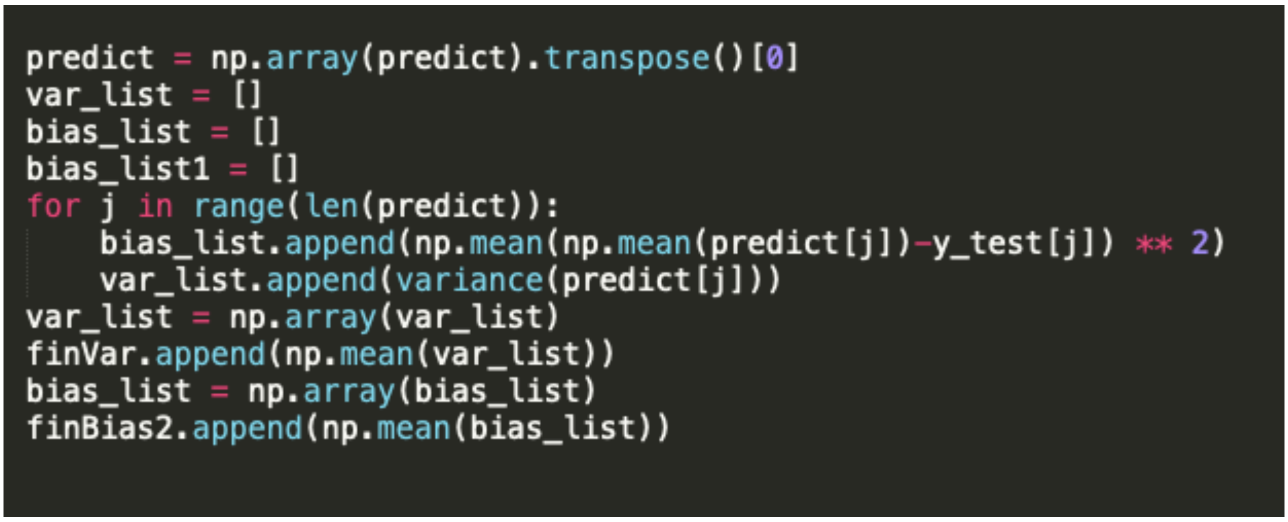
```
file = open('data.pkl', 'rb')

data = pickle.load(file)

file.close()

data = data.transpose()
x = data[0].reshape(-1,1)
y = data[1].reshape(-1,1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1)
x_train = np.array_split(x_train, 10)
y_train = np.array_split(y_train, 10)
```

Once training data and test data is made we use PolynomialFeatures to generate a feature matrix which consists polynomial combinations of the features with integer degree in range [1,9].



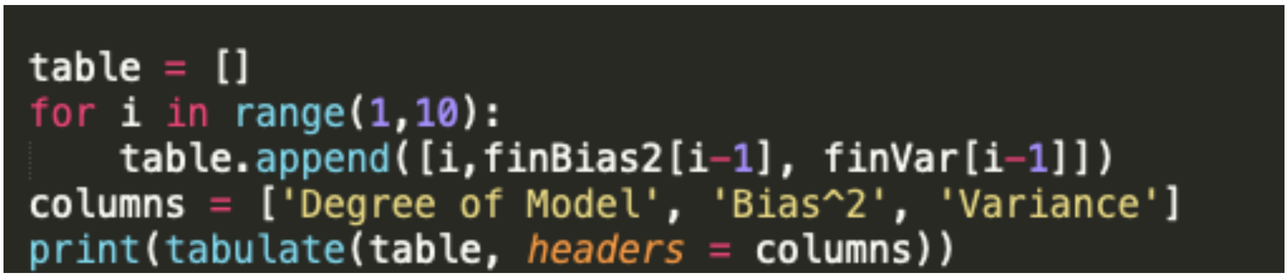
```python
for i in range(1,10):
    predict = []

    for j in range(10):
        poly = PolynomialFeatures(degree=i)
        lm = linear_model.LinearRegression()
        x_list = poly.fit_transform(x_train[j])
        lm.fit(x_list, y_train[j])
        x_test_poly = poly.fit_transform(x_test)
        predict.append(lm.predict(x_test_poly))
```

One matrices are filled, we use these values to fit into a model and the values are stored in numpy array which are then used to make Variance and Bias calculations.



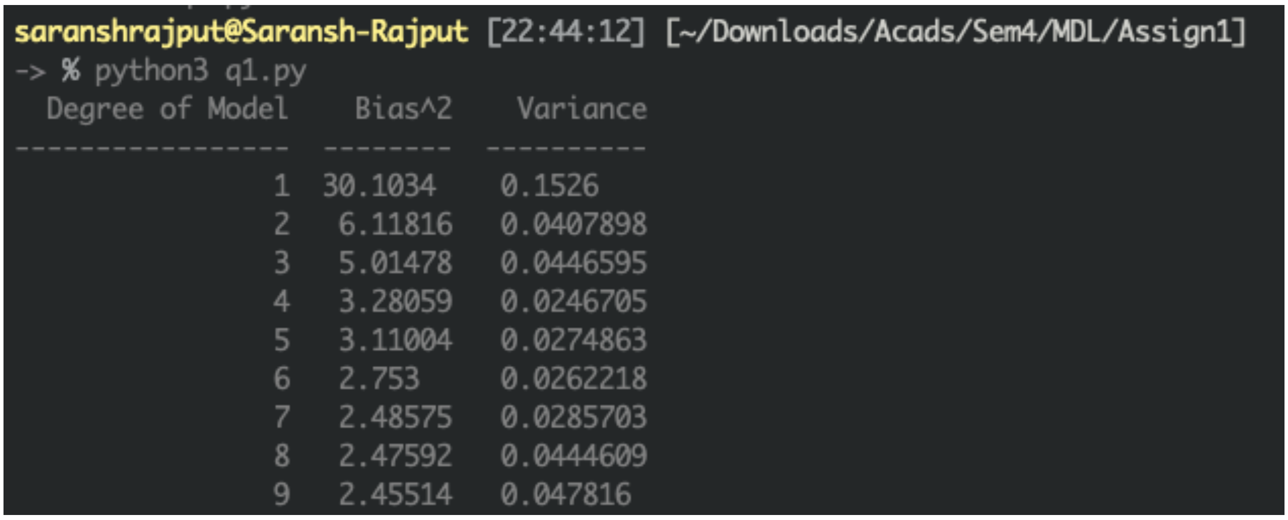
```python
predict = np.array(predict).transpose()[0]
var_list = []
bias_list = []
bias_list1 = []
for j in range(len(predict)):
    bias_list.append(np.mean(np.mean(predict[j])-y_test[j]) ** 2)
    var_list.append(variance(predict[j]))
var_list = np.array(var_list)
finVar.append(np.mean(var_list))
bias_list = np.array(bias_list)
finBias2.append(np.mean(bias_list))
```

Representing the data in the form of table:



```python
table = []
for i in range(1,10):
    table.append([i,finBias2[i-1], finVar[i-1]])
columns = ['Degree of Model', 'Bias^2', 'Variance']
print(tabulate(table, headers = columns))
```

Results:



```
saranshrajput@Saransh-Rajput [22:44:12] [~/Downloads/Acads/Sem4/MDL/Assign1]
-> % python3 q1.py
  Degree of Model    Bias^2    Variance
-----------------  --------  ----------
                1  30.1034    0.1526
                2   6.11816   0.0407898
                3   5.01478   0.0446595
                4   3.28059   0.0246705
                5   3.11004   0.0274863
                6   2.753     0.0262218
                7   2.48575   0.0285703
                8   2.47592   0.0444609
                9   2.45514   0.047816
```

Observations:

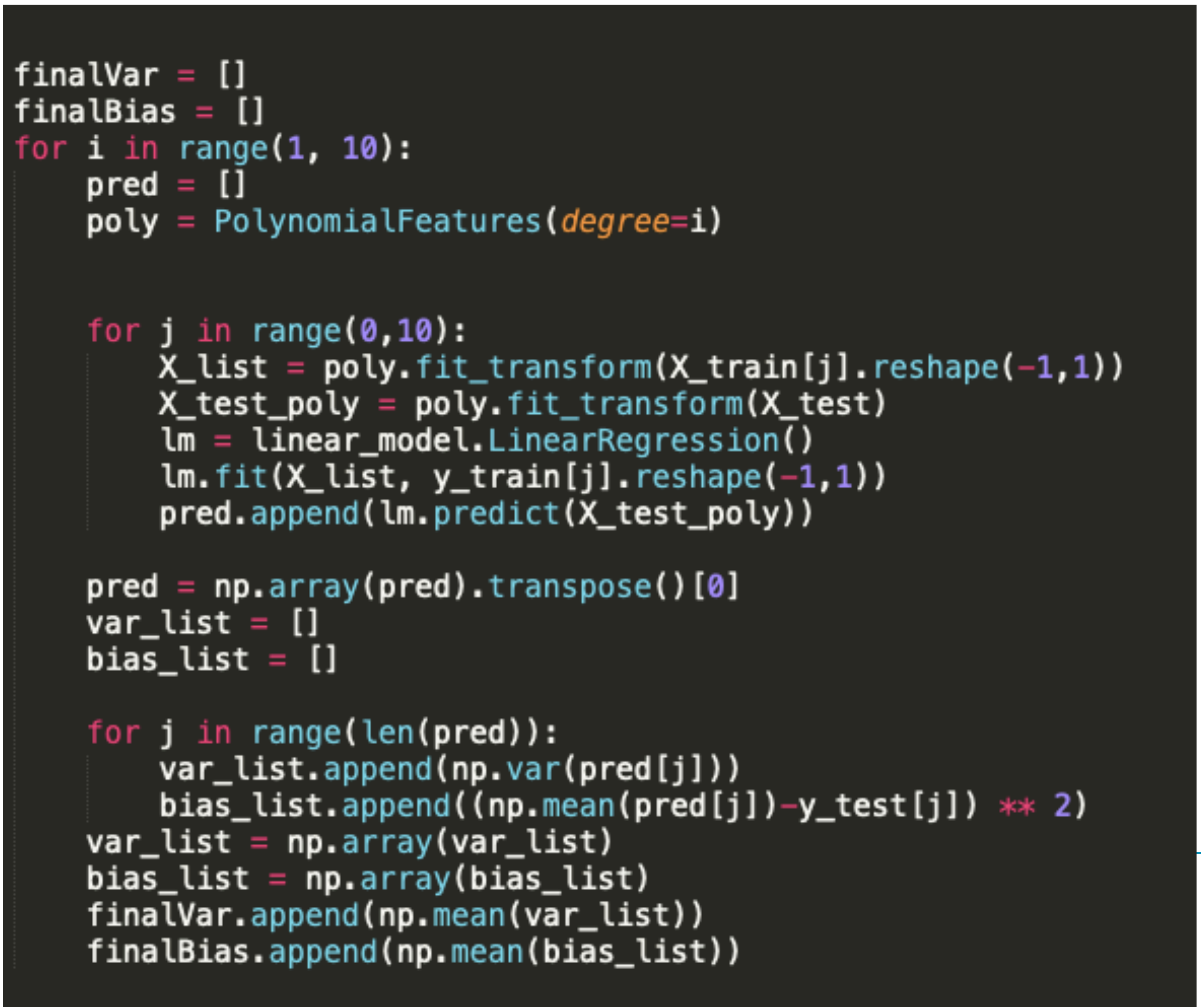
There is a huge drop in the bias^2 value as we move from 1st degree to second degree model following which there is a steady decrease in bias^2 value as the degree on model is increased. Expected that this trend continues upto a point, reaches a minimum and begins increasing again. This is when the model becomes an overfit.
For variance, it decreases till 4th degree model after which it begins increasing.

Question 2

Task: Use sklearn.linear_model.LinearRegression().fit() with the default parameters to fit the model to the data.

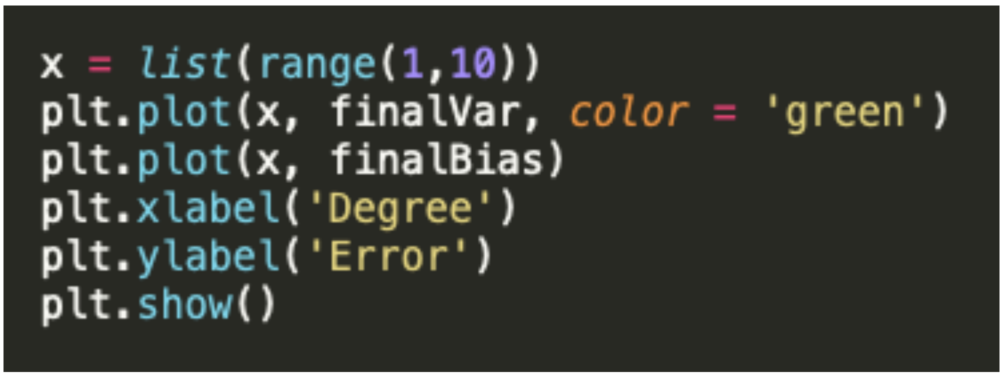
Similar to Question1, data is loaded as pickle. Here, we do not have to split given data into test and training but is already given as is.

Similar to what's done above, PolynomialFeatures are used to fill the feature matrices for different degrees in [1,9] which are then used to fit the model and the values are stored in a numpy array.



```
finalVar = []
finalBias = []
for i in range(1, 10):
    pred = []
    poly = PolynomialFeatures(degree=i)


    for j in range(0,10):
        X_list = poly.fit_transform(X_train[j].reshape(-1,1))
        X_test_poly = poly.fit_transform(X_test)
        lm = linear_model.LinearRegression()
        lm.fit(X_list, y_train[j].reshape(-1,1))
        pred.append(lm.predict(X_test_poly))

    pred = np.array(pred).transpose()[0]
    var_list = []
    bias_list = []

    for j in range(len(pred)):
        var_list.append(np.var(pred[j]))
        bias_list.append((np.mean(pred[j])-y_test[j]) ** 2)
    var_list = np.array(var_list)
    bias_list = np.array(bias_list)
    finalVar.append(np.mean(var_list))
    finalBias.append(np.mean(bias_list))
```

Matplotlib is used to represent the results graphically.

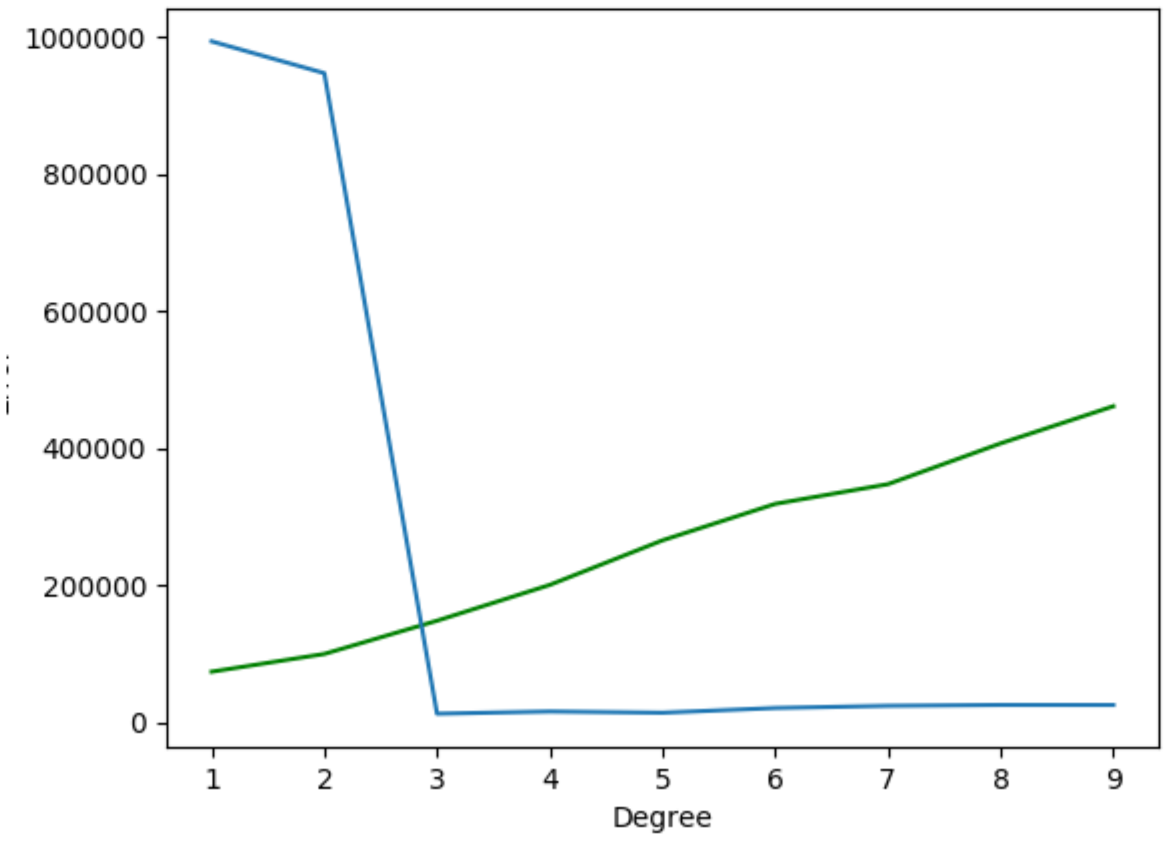


```
x = list(range(1,10))
plt.plot(x, finalVar, color = 'green')
plt.plot(x, finalBias)
plt.xlabel('Degree')
plt.ylabel('Error')
plt.show()
```

Results:

Variance 🟩

Bias^2 🟦



Observations:

Bias decreases from the 1st to 2nd degree, decreases further rapidly till the 3rd degree model, then sort of stabilises.
The variance continues to increase at an almost steady pace as the degree of model is increased.
The model is underfit(high bias and low variance) for degrees 1 and 2 and becomes overfit(low bias and high variance) as degree increases as can be inferred from the graph.
The training data set is closely set as a degree of 3 is enough to get a low bias and any higher degree provides negligible improvement.
But the test data set is not that similar to the training set as the higher the degree the more the model fits the training set but the variance also increases hence showing significant difference in the test and training set.