# Module 2 Homework

ISE-529 Predictive Analytics

*Note: In answering the following questions, you may only use functionality from Base Python, NumPy, Pandas, or Seaborn*

## 1. Evaluating regression functions

1A) The file "HW 2 Problem 1 Data.xlsx" contains two datasets - training and test in two different spreadsheet tabs. Read the tables into two dataframes (training_data and test_data) and display the first 10 rows of each dataframe. Hint - look up the Pandas function for reading in Excel files.

In [136...
```
import pandas;

train_df = pandas.read_excel("HW 2 Problem 1 Data.xlsx", sheet_name = "Training Data");
test_df = pandas.read_excel("HW 2 Problem 1 Data.xlsx", sheet_name = "Test Data");
```

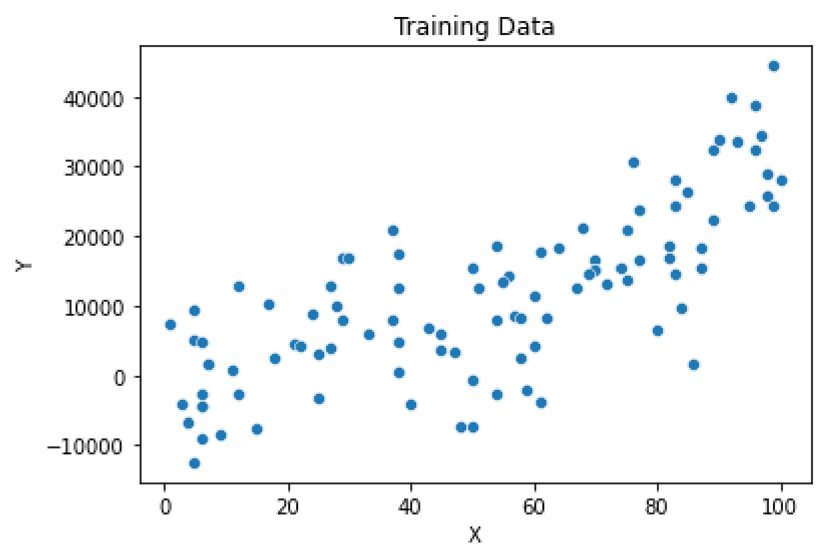
In [137...
```
print("Training Data");
print(train_df.head(10));
print("Test Data");
print(test_df.head(10));
```

```
Training Data
    X             Y
0  61  17661.067682
1  87  15482.455058
2  38  17444.767982
3   6  -4270.225550
4  54   8075.733045
5  29  16820.129406
6  77  23921.367914
7  70  16541.631267
8  21   4425.240897
9  76  30681.044509
Test Data
    X             Y
0  36  15986.579536
1  64  -2761.487999
2  66   9752.039830
3  88  20038.697197
4   4 -13421.192312
5  80   6644.121448
6  22 -10124.906855
7  85  21502.561217
8  63  12105.189261
9   8 -12161.603294
```

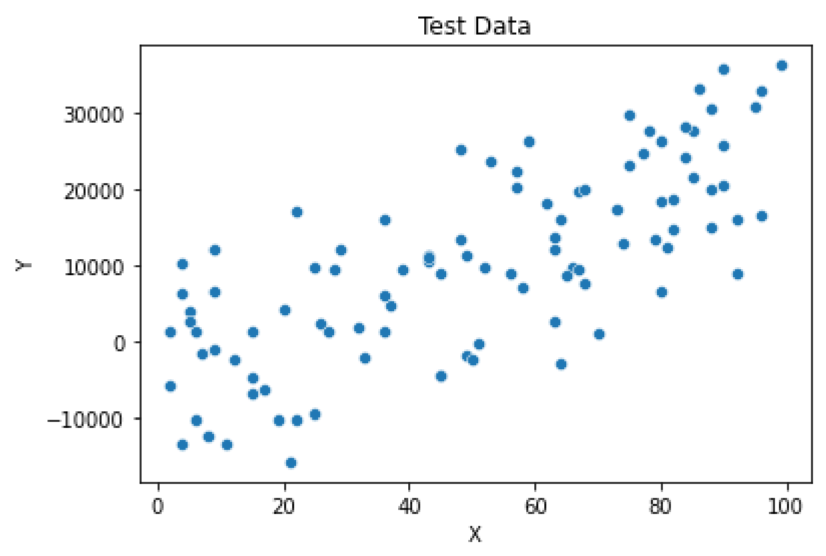
1b) Using Seaborn, create scatterplots of the two dataframes. For full credit, include a title ("Training Data" and "Test Data") for the two scatterplots.

In [138...
```
import seaborn;

seaborn.scatterplot(data = train_df, x = "X", y = "Y").set(title = "Training Data");
```



In [139...
```
seaborn.scatterplot(data = test_df, x = "X", y = "Y").set(title = "Test Data");
```



1c) Now, we are going to calculate the test and training MSEs for three different candidate regression models:

- F1: $300x - 4280$
- F2: $3.37x^2 - 36.8x + 1220$
- F3: $0.1x^3 - 12x^2 + 590x - 3600$

Create three functions for these three candidate models - f1(), f2(), and f3(). They should take as an input $x$ and return the predicted value of $y$ corresponding to that input:

In [140…
```
def f1(x):
    return 300 * x - 4280;
```

In [141…
```
def f2(x):
    return 3.37 * pow(x, 2) - 36.8 * x + 1220;
```

In [142…
```
def f3(x):
    return 0.1 * pow(x, 3) - 12 * pow(x, 2) + 590 * x - 3600;
```

1d) Now, write a function calc_mse() that takes three parameter inputs:

- $x$ - an array (or Pandas series) of predictor $X$ values
- $y$ - an array (or Pandas series) of response $Y$ values
- $f$ - a function to be called for each value of the $x$ and $y$ arrays

The function should calculate the MSE for the function $f$ using the $x$ and $y$ data arrays

In [143…
```
def calc_mse(x_list, true_y_list, f):
    mse = 0;
    for idx in range(len(x_list)):
        expect_y = f(x_list[idx]);
        mse += pow(true_y_list[idx] - expect_y, 2);
    mse /= len(x_list);
    return mse;
```

1e) Call this calc_mse function six times to calculate the training and test MSE for each of the three models:

In [144…
```
print(calc_mse(train_df["X"].tolist(), train_df["Y"].tolist(), f1));
```
```
65116445.47500964
```

In [145…
```
print(calc_mse(train_df["X"].tolist(), train_df["Y"].tolist(), f2));
```
```
57922250.15746113
```

In [146…
```
print(calc_mse(train_df["X"].tolist(), train_df["Y"].tolist(), f3));
```
```
54103936.344325066
```

In [147…
```
print(calc_mse(test_df["X"].tolist(), test_df["Y"].tolist(), f1));
```
```
67828868.62264524
```

In [148…
```
print(calc_mse(test_df["X"].tolist(), test_df["Y"].tolist(), f2));
```
```
67805434.17472021
```

In [149…
```
print(calc_mse(test_df["X"].tolist(), test_df["Y"].tolist(), f3));
```
```
71191752.34205785
```

1f) Which of the three models would you select for use and why?

First of all, the lower MSE the better and 0 means the model is perfect. I would select f2 model, because it has the lowest MSE in test dataset and the second lowest MSE in train dataset. Even though f2 model doesn't have the lowest MSE in train dataset, I believe it may due to the overfitting of f3 model.

1g) Insted of writing functions, write a single line of Python code to calculate the test MSE for function F1. Hint: use the Python map and lambda functions.

In [150…
```
test_f1_mse = sum(
    map(
        lambda true_y, expect_y: pow(true_y - expect_y, 2),
        test_df["Y"].tolist(),
        map(
            lambda x: 300 * x - 4280,
            test_df["X"].tolist()
        )
    )
) / len(test_df["X"].tolist());

print(test_f1_mse);
```

```
67828868.62264524
```

## 2. KNN and Calculate Misclassification Rates

2a) Read the file "HW 2 Problem 2 Data.csv" into a dataframe called knn_data and display its first 10 rows

In [151…
```
knn_data = pandas.read_csv(filepath_or_buffer = "HW 2 Problem 2 Data.csv");
print(knn_data.head(10));
```

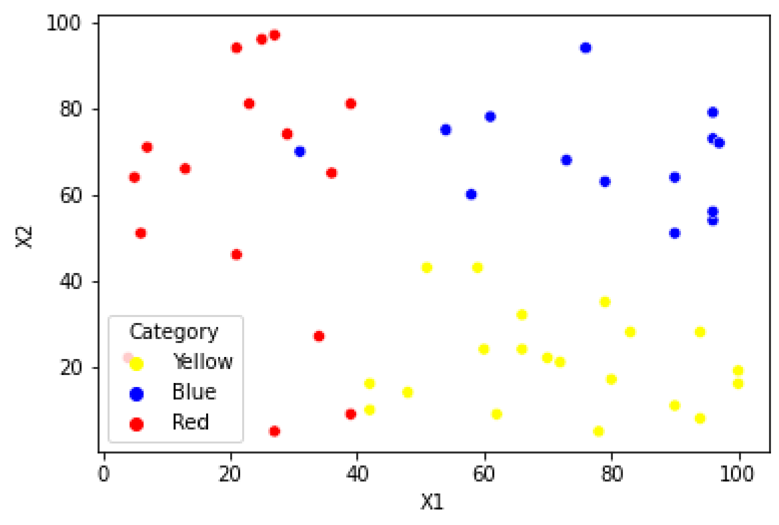
```
    X1  X2 Category
0   59  43   Yellow
1   60  24   Yellow
2   54  75     Blue
3   27   5      Red
4   96  73     Blue
5   51  43   Yellow
6   21  94      Red
7  100  19   Yellow
8   73  68     Blue
9   62   9   Yellow
```

2b) Using Seaborn, create a color-coded scatterplot of the data (where each point is colored with its category color). For full credit, be sure to color the points correctly with yellow, blue, and red colors.

In [152…
```
import seaborn;

color_dict = dict(
    {
        "Yellow": "yellow",
        "Blue": "blue",
        "Red": "red"
    }
)

seaborn.scatterplot(data = knn_data, x = "X1", y = "X2", hue = "Category", palette = color_dict);
```



Now, we are going to create and assess predictions for the category of each observation using KNN-1 and KNN-3 algorithms. This means for each observation we are going to determine a prediction for its category color as if we didn't know what it was and then we will be checking to see if the prediction matches its actual category color.

The problem sub-parts below will step you through the process of doing this.

2c) Create a two-dimensional numpy array of size 50x50 that has the distance of each observation to every other observation. Use a Euclidean formula to calculate the distances. Display the first row of the distance table (this row will show the distance of each observation from the first observation)

In [153…
```
import numpy;
import math;

def euclidean_distance(x1, y1, x2, y2):
    return math.sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));

all_list = [];
dist_list = [];
for idx1, row1 in knn_data.iterrows():
    x1 = row1["X1"];
    y1 = row1["X2"];
    all_temp_list = [];
    dist_temp_list = [];
    for idx2, row2 in knn_data.iterrows():
        x2 = row2["X1"];
        y2 = row2["X2"];
        dist_temp_list.append(euclidean_distance(x1, y1, x2, y2));
        all_temp_list.append((euclidean_distance(x1, y1, x2, y2), row2["Category"], idx2));
    dist_list.append(dist_temp_list);
    all_list.append(all_temp_list);

dist_array = numpy.array(dist_list);
print(dist_array[0]);
```

```
[ 0.         19.02629759 32.38826948 49.67896939 47.63402146  8.
 63.60031446 47.50789408 28.65309756 34.13209633 32.01562119 62.76941931
 57.93962375 17.02938637 38.07886553 33.42154993 37.44329045 39.44616585
 51.623638   28.3019434  42.48529157 58.87274412 38.60051813 28.28427125
 31.82766093 43.13930922 21.54065923 44.55333882 23.70653918 38.11823711
 31.01612484 59.05929224 38.89730068 35.05709629 29.68164416 13.03840481
 20.24845673 42.94182111 49.09175083 39.2173431  51.42956348 53.60037313
 62.96824597 52.34500931 31.90611227 25.55386468 49.49747468 37.12142239
 47.80167361 53.75872022]
```

2d) Now create a dataframe with 50 rows (one per observation) and 3 columns labeled 'nn1_cat', 'nn2_cat', and 'nn3_cat'. Populate the dataframe with the category color for the first, second, and third nearest neighbor for each observation.

Display the first ten rows of this dataframe

In [154…
```
nn1_list = [];
nn2_list = [];
nn3_list = [];

nn1_cat = [];
nn2_cat = [];
nn3_cat = [];

for temp_list in all_list:
    temp_list.sort();
    nn1_cat.append(temp_list[1][1]);
    nn2_cat.append(temp_list[2][1]);
    nn3_cat.append(temp_list[3][1]);
    nn1_list.append(temp_list[1]);
    nn2_list.append(temp_list[2]);
    nn3_list.append(temp_list[3]);

knn_cat = pandas.DataFrame(
    {
        "nn1_cat": nn1_cat,
        "nn2_cat": nn2_cat,
        "nn3_cat": nn3_cat
    }
)

print(knn_cat.head(10));
```

```
  nn1_cat nn2_cat nn3_cat
0  Yellow  Yellow    Blue
1  Yellow  Yellow  Yellow
2    Blue    Blue     Red
3     Red  Yellow  Yellow
4    Blue    Blue    Blue
5  Yellow    Blue  Yellow
6     Red     Red     Red
7  Yellow  Yellow  Yellow
8    Blue    Blue    Blue
9  Yellow  Yellow  Yellow
```

2e) Create a Pandas series nn1_preds that has the prediction for each observation using a KNN1 algorithm. Display the first 10 rows of the series.

In [155…
```
nn1_preds = pandas.Series(data = nn1_cat);
print(nn1_preds.head(10));
```

```
0    Yellow
1    Yellow
2      Blue
3       Red
4      Blue
5    Yellow
6       Red
7    Yellow
8      Blue
9    Yellow
dtype: object
```

2f) Calculate the misclassification rate for the KNN1 algorithm on this dataset

In [156…
```
knn1_miss_rate = 0;

for idx in range(len(nn1_list)):
    true_color = all_list[idx][0][1];
    predict_color = nn1_cat[idx];
    if true_color != predict_color:
        knn1_miss_rate += 1;
```

```
knn1_miss_rate /= len(nn1_list);
print(knn1_miss_rate);
```

```
0.12
```

2g) Create a Pandas series nn3_preds that has the prediction for each observation using a KNN3 algorithm. Display the first 10 rows of the series. (If the three nearest neighbors all have different color categories, use the first nearest neighbor category as the prediction)

In [157…

```
def most_color(color_list):

    color_map = [0, 0, 0];
    max_count = 0;
    max_color = "";

    for color in color_list:
        update_count = 0;
        update_color = "";
        if color == "Red":
            color_map[0] += 1;
            update_count = color_map[0];
            update_color = "Red";
        elif color == "Yellow":
            color_map[1] += 1;
            update_count = color_map[1];
            update_color = "Yellow";
        else:
            color_map[2] += 1;
            update_count = color_map[2];
            update_color = "Blue";

        if update_count > max_count:
            max_count = update_count;
            max_color = update_color;

    if max_count > 1:
        return max_color;

    return color_list[0];
```

In [158…

```
nn3_color_list = [];

for idx in range(len(nn1_list)):
    color_list = [
        nn1_list[idx][1],
        nn2_list[idx][1],
        nn3_list[idx][1]
    ];

    nn3_color_list.append(most_color(color_list));

nn3_preds = pandas.Series(data = nn3_color_list);
print(nn3_preds.head(10));
```

```
0    Yellow
1    Yellow
2      Blue
3    Yellow
4      Blue
5    Yellow
6       Red
7    Yellow
8      Blue
9    Yellow
dtype: object
```

2h) Calculate the misclassification rate for the KNN3 algorithm on this dataset

In [159…

```
knn3_miss_rate = 0;

for idx in range(len(nn3_list)):
    true_color = all_list[idx][0][1];
    predict_color = nn3_color_list[idx];
    if true_color != predict_color:
        knn3_miss_rate += 1;

knn3_miss_rate /= len(nn3_list);
print(knn3_miss_rate);
```

```
0.08
```