Project Update (11.17)

What I have done so far:

1.data preprocessing and feature selection

First of all I deleted the empty part of the dataset.

```
In [3]: pd.isnull(df)
        print (np.sum(pd.isnull(df)))
```

```
Formatted Date               0
Summary                      0
Precip Type                517
Temperature (C)              0
Apparent Temperature (C)     0
Humidity                     0
Wind Speed (km/h)            0
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         0
Daily Summary                0
dtype: int64
```

```
In [4]: df = df.dropna()
```

Then I set all the cloudy weather as label 1 and all the other weather as label "-1"

```
In [5]: df['Summary'].loc[df['Summary'] == 'Partly Cloudy'] = 1
        df['Summary'].loc[df['Summary'] == 'Mostly Cloudy'] = 1
        df['Summary'].loc[df['Summary'] != 1] = -1
        print(df.groupby('Summary')['Summary'].count())
```

```
Summary
-1    36387
 1    59549
Name: Summary, dtype: int64
```

Next I set the precipitation type "rain" to 1 and "rain" to 2 otherwise they can't be calculated mathematically.

```
In [7]: df['Precip Type'].loc[df['Precip Type'] == 'rain'] = 1
        df['Precip Type'].loc[df['Precip Type'] == 'snow'] = 2
        print(df.groupby('Precip Type')['Precip Type'].count())
```

```
Precip Type
1    85224
2    10712
Name: Precip Type, dtype: int64
```

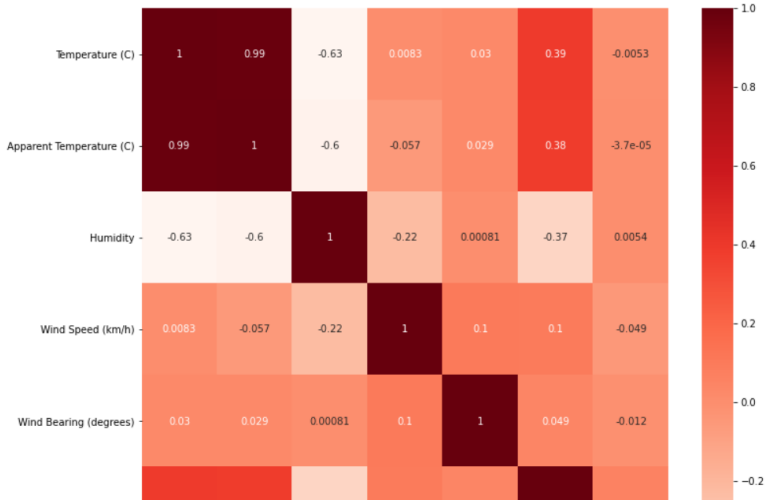
Next I deleted the column "Formatted date" "Daily summary" "Loud cover" since some of them are all 0 and some of their features are too much to use.

```
In [10]: df = df.drop(columns = ["Formatted Date"])
         df = df.drop(columns = ["Daily Summary"])
         df = df.drop(columns = ["Loud Cover"])
```

Next I get the heatmap of the existing features and it is shown that “temperature” and “apparent temperature” are highly correlated so I drop the column “apparent temperature”

```
In [13]: cor = X.corr()

         plt.figure(figsize = (12,12))
         sns.heatmap(cor, annot = True, cmap = plt.cm.Reds)
         plt.show()
```



Until now I have the dataset I want and I set the 70% of it as training set while 30% of it as testing set.

Algorithm design

I have trained and tested the linear regression

linear classifier

```
In [18]: w = np.linalg.inv(X_train.transpose()@X_train)@X_train.transpose()@y_train
         y_hat = np.sign(X_train@w)

In [19]: error_rate = np.sum(y_hat != y_train)/y_train.shape
         accuracy = 1 - error_rate
         accuracy

Out[19]: array([0.72343087])
```

SVD

```
In [24]: error_rate = np.sum(y_hat_svd != y_train)/y_train.shape
         accuracy = 1 - error_rate
         accuracy

Out[24]: array([0.71610453])
```

Regularization

```
lambda = 0.11980628596760347 , accuracy = [0.72344576]
lambda = 0.15898237887196842 , accuracy = [0.72343087]
lambda = 0.21096887018620014 , accuracy = [0.72343087]
lambda = 0.27995470003304446 , accuracy = [0.72343087]
lambda = 0.37149857228423705 , accuracy = [0.72343087]
lambda = 0.4929768608740504 , accuracy = [0.72344576]
lambda = 0.6541779793740121 , accuracy = [0.72346065]
lambda = 0.8680911066274185 , accuracy = [0.72344576]
lambda = 1.151952821962495 , accuracy = [0.72344576]
lambda = 1.5286359852052918 , accuracy = [0.72341598]
lambda = 2.0284927739346554 , accuracy = [0.72340109]
lambda = 2.6918003852647128 , accuracy = [0.72344576]
lambda = 3.5720064706253014 , accuracy = [0.72341598]
lambda = 4.74003581247511 , accuracy = [0.72335641]
lambda = 6.290005264075964 , accuracy = [0.72335641]
lambda = 8.346807447736152 , accuracy = [0.7233862]
lambda = 11.076174286766435 , accuracy = [0.72349043]
lambda = 14.698031265127607 , accuracy = [0.72344576]
lambda = 19.504218467271617 , accuracy = [0.72344576]
lambda = 25.88200631479307 , accuracy = [0.72341598]
lambda = 34.34530083853678 , accuracy = [0.72325218]
lambda = 45.57605292814517 , accuracy = [0.72325218]
lambda = 60.479208211750866 , accuracy = [0.72319261]
lambda = 80.2556252882861 , accuracy = [0.72314794]
lambda = 106.49883787933477 , accuracy = [0.72296925]
lambda = 141.32345775024794 , accuracy = [0.72243318]
lambda = 187.53556478348744 , accuracy = [0.72241829]
lambda = 248.85881380580645 , accuracy = [0.72243318]
lambda = 330.23447728612325 , accuracy = [0.72194178]
lambda = 438.21960058661415 , accuracy = [0.72119723]
lambda = 581.5153521111805 , accuracy = [0.72006552]
lambda = 771.6681414713509 , accuracy = [0.71932097]
lambda = 1024.0 , accuracy = [0.71777232]
```

SVD for Regularization

```
lambda = 0.15898237887196842 , accuracy = [0.66459683]
lambda = 0.21096887018620014 , accuracy = [0.66458194]
lambda = 0.27995470003304446 , accuracy = [0.66459683]
lambda = 0.37149857228423705 , accuracy = [0.66459683]
lambda = 0.4929768608740504 , accuracy = [0.66459683]
lambda = 0.6541779793740121 , accuracy = [0.66459683]
lambda = 0.8680911066274185 , accuracy = [0.66459683]
lambda = 1.151952821962495 , accuracy = [0.66459683]
lambda = 1.5286359852052918 , accuracy = [0.66462661]
lambda = 2.0284927739346554 , accuracy = [0.66459683]
lambda = 2.6918003852647128 , accuracy = [0.66462661]
lambda = 3.5720064706253014 , accuracy = [0.66461172]
lambda = 4.74003581247511 , accuracy = [0.66461172]
lambda = 6.290005264075964 , accuracy = [0.66458194]
lambda = 8.346807447736152 , accuracy = [0.66465639]
lambda = 11.076174286766435 , accuracy = [0.66484997]
lambda = 14.698031265127607 , accuracy = [0.66499888]
lambda = 19.504218467271617 , accuracy = [0.66526692]
lambda = 25.88200631479307 , accuracy = [0.66540094]
lambda = 34.34530083853678 , accuracy = [0.66589234]
lambda = 45.57605292814517 , accuracy = [0.66621994]
lambda = 60.479208211750866 , accuracy = [0.66651776]
lambda = 80.2556252882861 , accuracy = [0.66681558]
lambda = 106.49883787933477 , accuracy = [0.66730698]
lambda = 141.32345775024794 , accuracy = [0.6677686]
lambda = 187.53556478348744 , accuracy = [0.66849825]
lambda = 248.85881380580645 , accuracy = [0.66997245]
lambda = 330.23447728612325 , accuracy = [0.67116373]
lambda = 438.21960058661415 , accuracy = [0.67295064]
lambda = 581.5153521111805 , accuracy = [0.6741568]
lambda = 771.6681414713509 , accuracy = [0.67592882]
lambda = 1024.0 , accuracy = [0.67773062]
```

Up to now the highest accuracy is around 72% which means there are still a lot I need to improve.

My plan ahead

According to my plan, I give 10 days for finishing the report, but right now it seems like I need more time to finish the KNN and neural network design, so I decide to set a new timeline:

Nov 18th – Dec.6th (used to be 1st)

Applying KNN and neural networks on the dataset and analyzing the result.

Dec.7th – Dec.12th

Report (Will start the introduction and early part of report from now on)