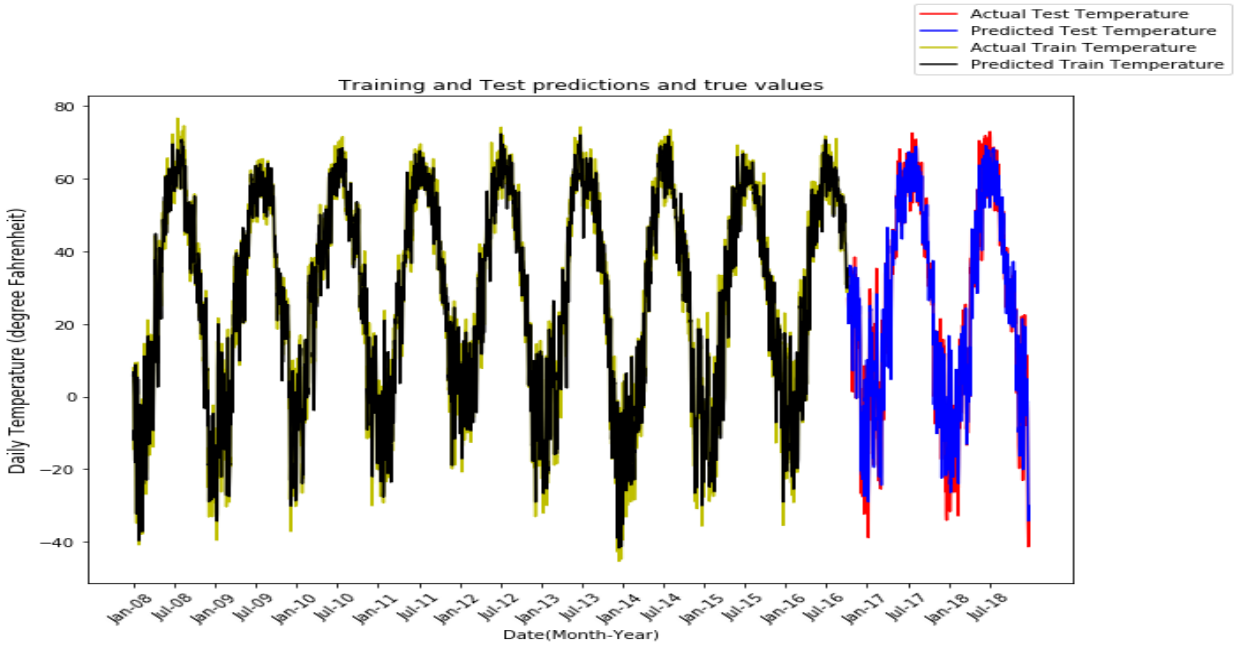
Problem 1

a) I'm predicting the new daily temperature of the station named STONY RAPIDS ARPT in canada which I got by setting the random seed = 30 (in the original notebook it was 3). Following are the mean squared errors on predicting the daily temperature obtained on the training, validation and test set using a Random Forest Regressor by setting following hyperparameters :- n_estimators = 100, max_depth = 10 and random_seed = 10. All other hyperparameters have default values. This is the initial model. I used the data processing cells from the notebook timeseries_prediction_TEMP.ipynb as it is without any change. The model is trained only on train data which is 60% of the complete data. This train data has been generated from train_test_split() function of sklearn with random_shuffle = True. In the following table, the % in the braces indicates the fraction of complete data.

Set	Mean Squared Error(mse)
Train(60%)	9.536931429588996
Validation(20%)	42.748009041567165
Test(20%)	46.220861705502315

b) Following is the plot of the model predictions(using trained random forest) and truth values on training(both train and validation set) and test data vs the month-year index, in predicting daily temperature for the station mentioned in part(a). In the following plot and all the subsequent plots, the training set and validation set are together referred to as train in the plots' titles and legends. This is because training and validation sets were generated from train_test_split() function by shuffling training data and as per the assignment instructions the data needs to be plotted sequentially. Therefore true and predicted values of both the training and validation sets are first sorted by indices and then plotted.



By looking at the mean squared errors, it is clear that the model isn't able to fit the training data properly owing to high mean squared error in training data. Hence, its generalization performance is also very poor on the test set. This might be due to the lesser number of trees or due to limiting the depth of the decision trees(in the random forest) which limits the model capacity to fit to the training data. Another factor that degrades the model's performance is the use of redundant and bad features which inhibits the model's capability to learn good patterns.

To improve this random forest regressor, I suggest that the hyperparameters of the random forest regressor should be properly tuned. It is very important to search for good values of the number of decision trees and maximum depth. We can do this using Randomized Search Cross Validation. Further, it is also necessary to select the good features from the dataset which allows the random forest regressor to learn the good patterns from the training set which makes prediction better. So to achieve this, we should perform feature selection. Numerical features that have high correlation with other features are redundant and should be removed from the training data. On this transformed training data, the random forest regressor should be trained and its hyperparameters be selected using a validation set.

Problem 2

(a) I'm predicting the Windspeed of the station named STONY RAPIDS ARPT in canada which I got by setting the random seed = 30 as in problem 1 part (a). Following are the mean squared errors obtained on the training, validation and test set using a

Random Forest Regressor model(n_estimators = 100, max_depth = 10 and random_seed = 10) similar to the one in problem 1 trained only on train data. I have used the same exact data processing cells given in *timeseries_prediction_Wind.ipynb* without any modification for this part.

Set	Mean Squared Error(mse)
Train(60%)	1.9576900507233845
Validation(20%)	4.720965045122381
Test(20%)	4.587456719465586

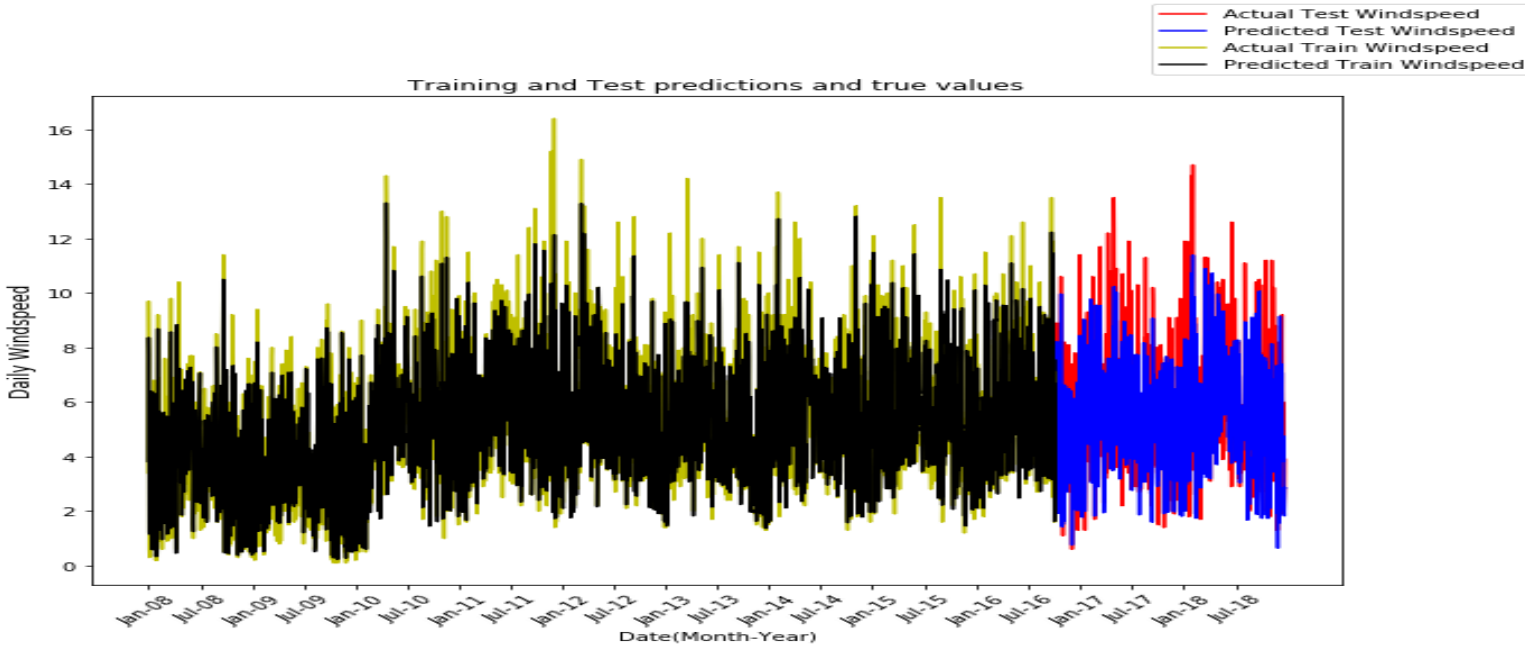
As compared to this model's performance on predicting daily temperature(problem 1), this model did a much better job in predicting the windspeed. This is evident from train, validation and test set errors.

(b) I improved the model based on the suggestions I made in problem 1 (b) part. Following are the predicted values on train, validation and test data -

Set	Mean Squared Error(mse)
Train(60%)	0.2543750546415981
Validation(20%)	1.9042181741001556
Test(20%)	2.0151539940550682

I added the maxspd feature in the training data so as to take all features into account to perform exploratory data analysis. It was not included in the original data processing steps.I performed the feature selection on my data and removed the following features - ['max_lag1', 'max_lag2', 'max_lag3', 'min_lag1', 'min_lag2', 'min_lag3']. This is because these features have very high correlation with temp_lag variables(~0.98) and all of them have similar correlation with windspeed. Then I created a grid of different values of hyperparameters of random forest regressor comprising of n_estimators, max_features, max_depth, bootstrap, criterion, min_samples_split and then carried out randomized search cross validation for 1000 iterations on predefined split created from the validation data. I created a new Random Forest Regressor using those good values of hyperparameters I obtained from search and trained it on the training set. I observe the performance improvement. The primary reason for the performance improvement is that now we have exhaustively searched for the right hyperparameter values which ensure good generalization performance(by giving it adequate amount of capacity) of the random forest regressor(n_estimatorws = 100, max_dept=80, min_samples_split = 3 etc). Moreover, by eliminating the highly correlated features, we reduce the redundancy in the training data which allows the model with current capacity to better fit the data with good features which enhance the performance. This approach makes sure that I'm keeping the most informative variables in my dataset and at the same time reducing the dimensionality of my training data by removing the redundant features.

Plot - Following is the plot of the training(both train and validation set) and test data, predictions and truth values from improved model described in part(b)



Problem 3

(1) In this problem, using the NOAA dataset, I'm predicting the Windspeed of the station named STONY RAPIDS ARPT in canada which I got by setting the random seed = 30 as in problem 1 part (a). I implemented a multilayer perceptron based regressor which has just one hidden layer of 100 units. I trained this model for 200 iterations on the training data after performing the feature selection similar to part b of problem 2. Default L2 Regularization has also been applied during training. I got the following mean squared errors reported on train, validation and test set

Set	Mean Squared Error(mse)
Train(60%)	1.5771495713095338
Validation(20%)	1.7565439431183483
Test(20%)	1.9893927854452491

It can be observed that the multilayer perceptron has training error higher than the random forest regressor but the validation and test errors are lower than what observed in improved random forest in problem 2 part (b). This can be explained in the following way. The multilayer perceptron has higher representational power/capacity to learn more complicated patterns from the data but when it is used with regularization, it's tendency to overfit on the training data reduces and its tendency to generalize gets better which is the reason why it reported lower validation and test error. I choose Multilayer perceptron because I wanted to use a parametrized model(which gives it very high representation power) which can learn efficient representation of patterns in the data which can then be exploited to predict the windspeed whereas random forest is an ensemble of decision trees and decision trees are non-parametric models which learn to predict by making divisions in the input space. These models have a high tendency to overfit the training data although ensembling reduces this but still there is a good tendency to overfit. Whereas regularization used in Multilayer perceptron is efficient in reducing overfitting resulting in higher training error but lower test and validation error. This is how machine learning answers the predictive questions by building parametric/non-parametric models and learning them using training data to recognize the patterns in the data which is used to perform the subsequent tasks such as classification, regression etc.

Following is the plot of the training(both train and validation set) and test data, predictions and truth values from multilayer perceptron in problem 3 -