Data-Driven Prediction of Weather Forecast Temperature

Shilpu Srivastava[1], Anusha Kankari[2], and Ankith Jain Kala[3]

[1] University of Ottawa 75 Laurier Ave E, Ottawa, ON K1N 6N5
ssriv071@uottawa.ca
[2] University of Ottawa 75 Laurier Ave E, Ottawa, ON K1N 6N5
akank047@uottawa.ca
[3] University of Ottawa 75 Laurier Ave E, Ottawa, ON K1N 6N5
akala066@uottawa.ca

Abstract. The main objective of this project was to explore data, discover interesting data patterns and built a model that predicts the weather forecast temperature with maximum correlation and minimum mean squared error. The weather forecast temperature dataset contains an amalgamation of indoor and outdoor attributes i.e. a total of 24 attributes with 4137 instances. The real-world data is usually inconsistent and likely encompasses with errors. Data preprocessing is one of the most crucial and proven methods that resolve all these issues. As part of data preprocessing, we applied various data mining techniques to handle missing values, check attribute correlations, feature selections, z-score normalization, principal component analysis, clustering and box-plots for outlier detection.
In data modeling, we constructed both machine and deep learning models to get a better understanding of which model suits best for this multivariate and time-series dataset. We implemented a linear, SVM, decision tree, k-nearest neighbor, gradient boosting and bagging machine learning models. Neural network and LSTM deep learning models are developed using Keras to understand which model among machine and deep learning predict better results for this time-series regression problem. In model evaluation, we considered Pearson, Spearman correlations and Mean squared error metrics to compare and select which model suits best for this dataset. This paper will conclude by stating the best model and provide some information about the future work that can be carried further on this dataset.

Keywords: neural network, LSTM, pearson, spearman, clustering, bagging, boosting, SVM, KNN.

1 Introduction

The weather forecast temperature dataset is taken from the UCI repository. We observed the data was collected every 15 minutes for selective days in 3 months March, April, May in the year 2012. This dataset contains a total of

24 attributes that include various Indoor(Indoor Temperature, Carbon dioxide, Relative Humidity, Lightning of dining room and living room) , Outdoor(Rain, Sun dusk, Wind, Sunlight, Sun irradiance, Enthalpic motor, Outdoor temperature, Outdoor Relative Humidity) and Time-series(Date, Time, Day of the Week) attributes with 4137 instances.

What was the major problem we investigated in this project?

This project's major aim was to analyze the data collected by various indoor and outdoor parameters and construct the best model that predicts the weather forecast temperature with minimum loss and maximum correlation value. We considered this as a regression problem and built various machine learning models to predict the "weather forecast temperature" target attribute. Deep learning models like neural networks and LSTM are very well known to provide great results for time-series data. So, we want to compare all these models and state the best model that predicts accurate results.

How have we examined the problem in a sequential manner?

The model prediction values are fully dependent on the training data. It is very important to make sure the data is consistent and meaningful before training the models.

The first step considered was preprocessing the data by applying various techniques to explore, clean and prepare data before fitting to the model. We tried to look for missing data, analyzed the attribute correlations using a heat map and performed feature selection to eliminate irrelevant attributes. We normalized the data to make sure they are in the same range and extracted the day, month, year, time values from the "Date" and "Time" attribute to solve this time-series problem. We reduced the dimensions using principal component analysis and performed clustering to see the distribution of data using k-means, agglomerative and gaussian clustering algorithms. Also, detected outliers using box-plots.

In model construction, we implemented linear regressor, support vector regressor, k-nearest neighbor, gradient boosting, bagging and decision tree machine learning models to examine this regression problem. As we discussed, this dataset is of time-series type so we even implemented neural network and LSTM deep learning models. For each model, we used the train and test split approach to fit and predict target results. To investigate more on model accuracy, we even plotted a graph between the predicted and test results.

For each model, we even calculated mean squared error, Pearson and Spearman coefficient to understand how accurate the model predictions are. Finally, among all the models we chose a model that possesses low mean squared error and high correlation coefficient values. This is the approach we considered to solve this time-series regression problem.

2 Data Preparation, Exploration and Transformation

Real-world data is generally incomplete, inconsistent and noisy. Data preprocessing includes various data mining techniques to clean, transform and reduce

raw data into a meaningful and consistent one. We aimed for accurate prediction from our models and applied various data mining techniques before implementing the models. In this section, we will explain how we explored and analyzed our dataset in a detailed manner.

2.1 Missing data

Missing data or values can be referred to as a data value not recorded for an attribute at a particular observation time. The missing values in a dataset can cause a lot of problems like biased results that may lead to invalid conclusions. We used "dataframe.isnull()" function to check if our dataset contains any missing values. We observed our dataset has zero missing values and plotted a graph refer to Fig 1 that explains each attribute has zero missing or null values.

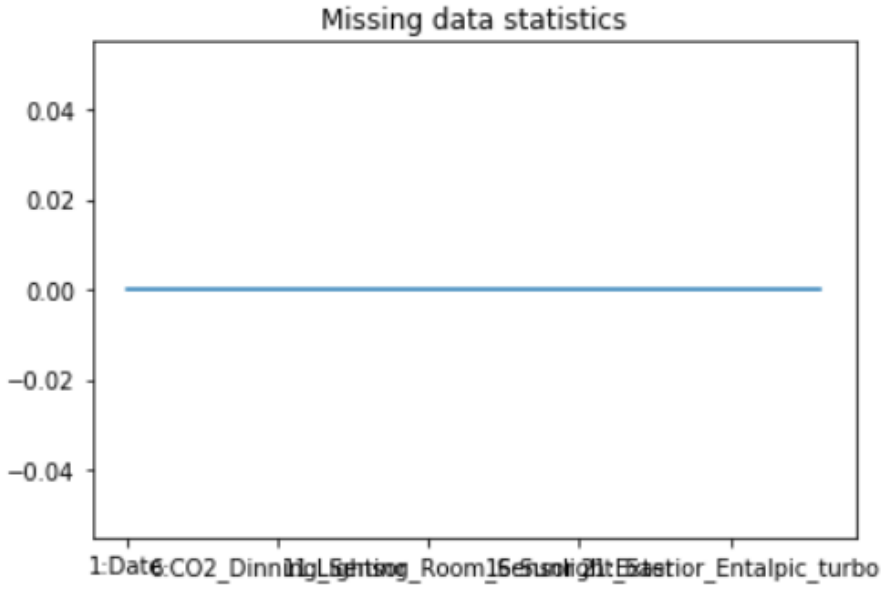

Fig. 1. Missing data statistics

2.2 Features Correlation

As part of data exploratory analysis, it is very important to understand the correlation of features with the target attribute. Features with high correlation value are highly dependent on the target variable and play an important role in prediction. We generated the correlation matrix and heatmap for our dataset refer to Fig 2 which explains darker the color, highly correlated the attributes

are. Feature selection can be performed using this correlation information which is discussed in Section 2.3.

Fig. 2. Features correlation using Heatmap

2.3 Feature selection

The final dataset included 24 features, it is important to understand the importance of each feature before performing any evaluation. This procedure of analyzing, finding the importance and meaning of each feature that adds value to our dataset is known as feature selection that shows a great impact on model performance. Using the feature correlation heatmap metrics from Fig 2, we defined threshold value of 0.3 and selected the important features that are more valid for our dataset. A bar chart of relevant features is plotted Refer to Fig 3, which gives information about the selected features and their values.

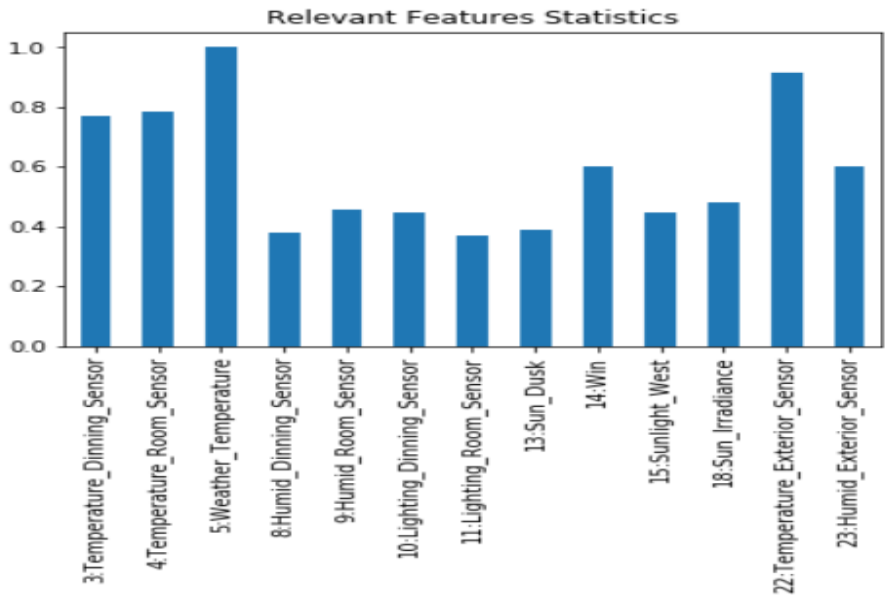

Fig. 3. Relevant features statistics

2.4 Normalization

The data in the attributes are observed to be in different ranges. Normalization is one such technique applied during data preparation to make sure the data in attributes come to a common scale. This affects the accuracy of models and we applied z-score normalization using StandardScaler() class. This calculates mean and standard deviation for each feature and scales them.

	3:Temperature_Dinning_Sensor	4:Temperature_Room_Sensor	5:Weather_Temperature	8:Humid_Dinning_Sensor	9:Humid_Room_Sensor	1
0	-0.088634	-0.104528	-0.478343	-1.498348	-1.734006	
1	-0.100924	-0.116610	-0.478343	-1.492249	-1.745903	
2	-0.126681	-0.147420	-0.478343	-1.488548	-1.731921	
3	-0.155881	-0.169984	-0.478343	-1.481895	-1.722435	
4	-0.181245	-0.195146	-0.478343	-1.471915	-1.707007	
...	...	...	...	...	...	
4132	0.198832	0.203181	-0.478343	-0.149404	-0.344494	
4133	0.165405	0.168927	-0.676453	-0.148669	-0.340637	
4134	0.133216	0.131895	-0.706922	-0.146452	-0.336141	
4135	0.109451	0.112955	-0.706922	-0.149778	-0.329704	
4136	0.081279	0.088791	-0.676453	-0.153839	-0.326498	

Fig. 4. Normalized dataset

2.5 Principal component analysis

Clustering is one of the important data analysis techniques that aim to group similar data into one and dissimilar one to others. This gives us a better understanding of the distribution and relationships of the data instances. Principal component analysis(PCA) is a technique that enables dimensionality reduction which transforms attributes of a dataset into fewer principal components. We plotted a graph between variance and features to identify the optimal value for the number of principal components refer to Fig 5.

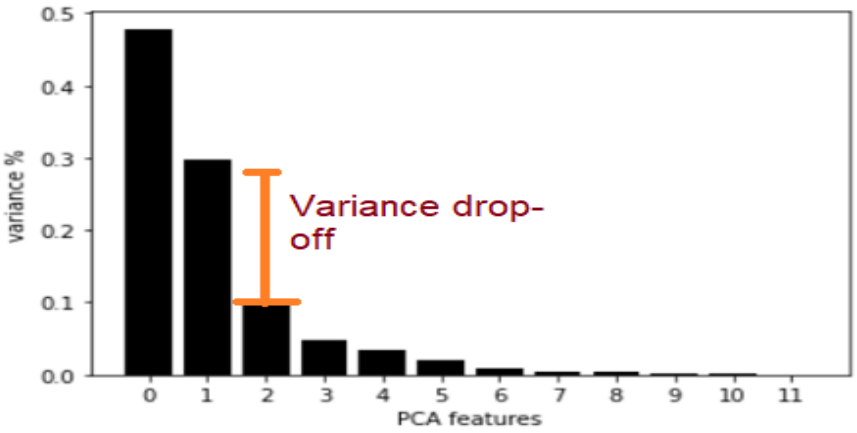

Fig. 5. Plot between variance and PCA features

As you can see, there is variance drop-off at value 2. So, we applied PCA on our dataset and reduced it into 2 principal components. We visualized these components on a bar chart refer to Fig 6.

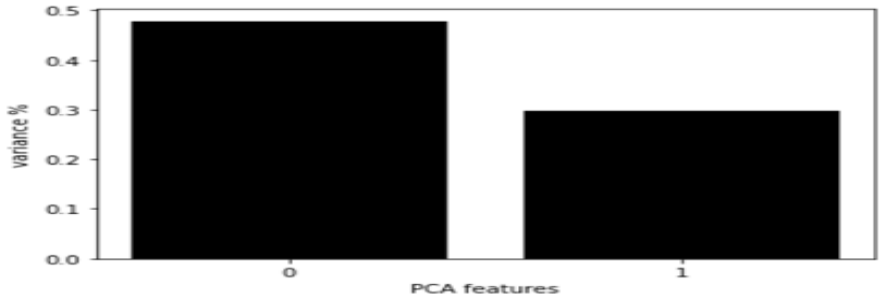

Fig. 6. Plot between variance and principal components

2.6 Clustering

Feature selection helped us to identify the relevant 13 attributes of our dataset. Later we applied k-means, agglomerative and gaussian clustering algorithms to observe the distribution of data. We observed that clusters are overlapping on each other refer to Fig 7, that led us to apply PCA discussed in section 2.5.

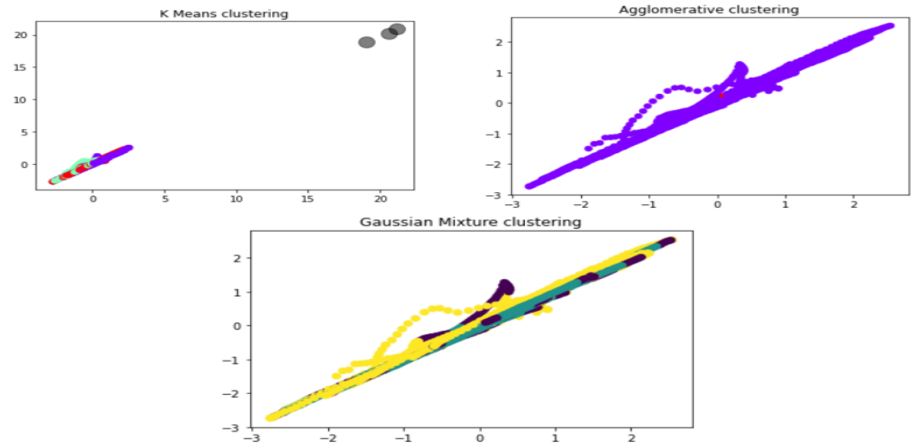

Fig. 7. Clustering with 13 attributes

We applied the same clustering algorithms on the two principal components and now the clusters look great refer to Fig 8 without overlapping and minimum intra and maximum inter distance between clusters.

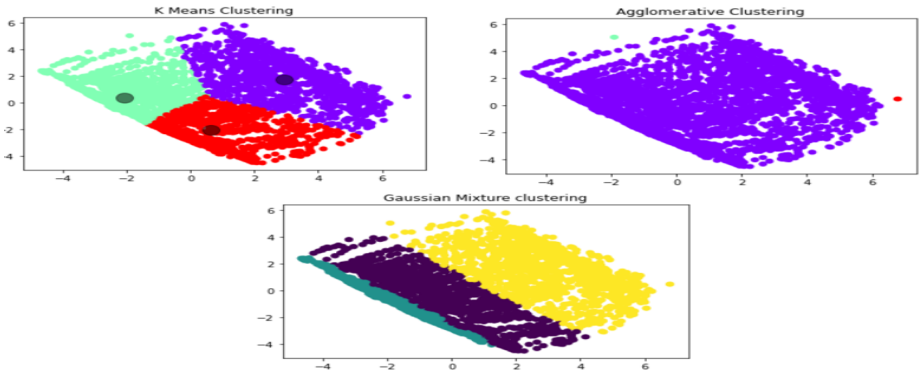

Fig. 8. Clustering after principal component analysis(PCA)

2.7 Outlier Detection

Outliers are the data instances with extreme behavior and deviate from normal data instances. It is very important with the quality of data to have accurate results. In the preprocessing, detecting these outliers and eliminating them will show an impact on the model performance. We applied box-plots refer Fig 8 on our dataset to detect the outliers and observed they were very minute outliers in 2 or 3 columns and we kind of neglected it since our model was performing way better than expected which will be discussed in later sections. We decided to consider eliminating outliers as a future work.

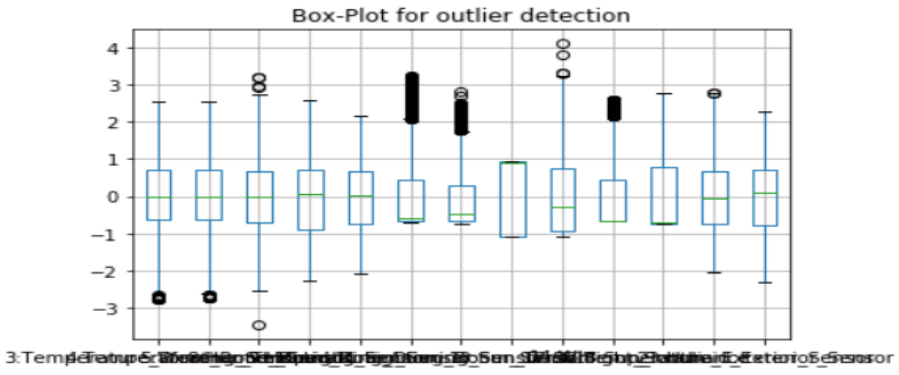

Fig. 9. Box-plot for outlier detection

3 Data Modeling

The major goal of this project is to build a model that predicts the weather forecast temperature with minimum loss and maximum correlation i.e accurate results. We considered this as a time-series regression problem and planned to construct different machine learning and deep learning models. In this section, we will discuss various algorithms we considered from different classifiers like linear, distance, tree and ensemble. Also, how did we choose the number of hidden layers, activation functions, epochs for neural network and the LSTM models.

Before constructing models, we preprocessed the data discussed in Section 2 and made it consistent enough to fit the model. Data model evaluation is the core part of our project that helps us to choose the best model to solves our problem. Before constructing and evaluating models we need to train and fit the data for it to predict results. In this project, we used the **train/test split** mechanism that splits our entire dataset into 70% training and 30% test datasets. We then fit the model with training data and predicted results on test

data. We used Pearsons and Spearman coefficients to check how accurate the results comparing with the test and predicted results discussed in Section 4.

3.1 Machine Learning models

3.1.1 Linear-based algorithms: This classifier separates input vectors into classes using different decision boundaries like line, plane, hyperplane based on dimensions. It is also stated that the separation decision is based on the linear combination value.

- **a. Linear Regression:** This algorithm tries to explain the linear relationship of the target variable on various independent variables. This algorithm tries to fit the data instances along with the decision boundaries that give fewer error results[1]. We imported the LinearRegression class, instantiated it to train and predict the results of the model.
- **b. Support vector machine:** This algorithm aims to find a hyperplane in an N-dimensional space to separate the data instances[2]. This algorithm takes linear and non-linear kernels as parameters. We observed that for our dataset a non-linear kernel radial basis function(RBF) is giving less mean squared error value.

3.1.2 Distance-based algorithms: This algorithm will group or classify the outcomes by measuring the distance. The distance can be calculated using various metrics like Euclidean, Minkowski, Manhattan, Chebychev, etc[3].

- **a. KNN Regression:** This is a non-probabilistic algorithm that uses distance metrics to classify the dependent target variables. The value of K is constant and this is used to predict the value for a test set based on the closest instance plotted based on the training data set. The drawback of this algorithm is the prediction of a target value depends on K value so, we plotted an elbow curve refer to Fig 9 between K values ranging from 1 to 20 against the models RMSE values to choose the best K-value.

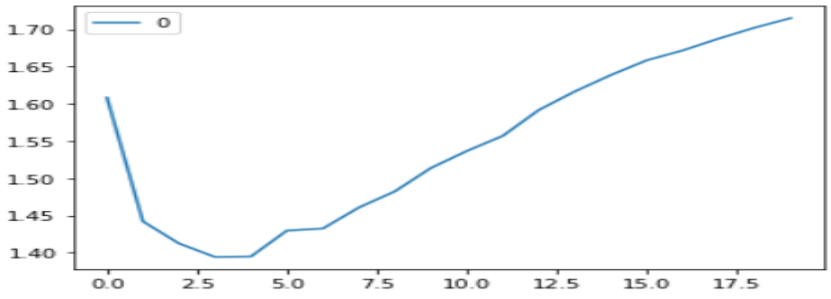

Fig. 10. RMSE vs K-values

3.1.3 Tree-based algorithm: Compare to linear and distance, the tree-based models empower the model prediction. They construct a tree using nodes where each node is connected by branches and the leaf node represents the target values. Various metrics are calculated to decide and split the nodes to reach the target value. We observed for our dataset the loss value and accuracy of the model are better than linear and distance-based models.

- **a.Decision tree Regression:** This algorithm splits the final dataset into two or more homogenous subsets by considering the underlying significant differentiator between the input features[5]. This algorithm is easy to understand and useful for data exploration. The drawback of this algorithm is they are more prone to overfit and if the depth of the tree is huge, it has a direct impact on model performance.

3.1.4 Ensemble- based algorithm: The main objective of this classifier is to train multiple models and integrate all the results into one model and increase overall performance. It uses various techniques like voting, weighted averaging and averaging to select one best final model[4]. Compare to individual model decision results, this classifier provides better predictions. For our dataset also, we observed this classifier based algorithm performed way better than above discussed classifiers.

- **a.Bagging:** It is very well known as Bootstrap Aggregating in which original data is equally divided into subsets with replacement and given to either heterogeneous or homogenous models and final prediction is determined by integrating results from all models using the voting technique. In our project, we used BaggingRegressor() class with input parameters like number jobs running parallelly as 4, the base estimator is decision trees with count of 10. The drawback is that it is computationally expensive for an enormous amount of real-time data.
- **b.Boosting:** It is a sequential process, where the following model updates the weight of the previous model. We used GradientBoostingRegressor() with a maximum depth of 4, base estimators as decision trees and a learning rate of 0.01.

3.2 Deep Learning models

This dataset is a time-series type, we wanted to explore how accurate these deep learning models on our dataset. We imported "Keras" an open-source python library that runs on top of the tensor flows to build these models.

3.2.1 Sequential neural network: We defined a sequential model with 1 input, 2 hidden and 1 output layer. The first hidden layer consists of 40 neurons and 'relu' as an activation function. The second hidden layer consists of 20 neurons and 'relu' as an activation function. Finally output layer with 'linear'

as an activation function. We manually tried tunning the network using various activation functions, different numbers of neurons at each layer. The above-specified combination gave us less loss value and great accuracy results.

We plotted a graph between the mean square error and the number of epochs, to define an optimal epochs value so that network gets trained well. Refer to Fig 11, based on this graph we can define the number of epochs value as 200.

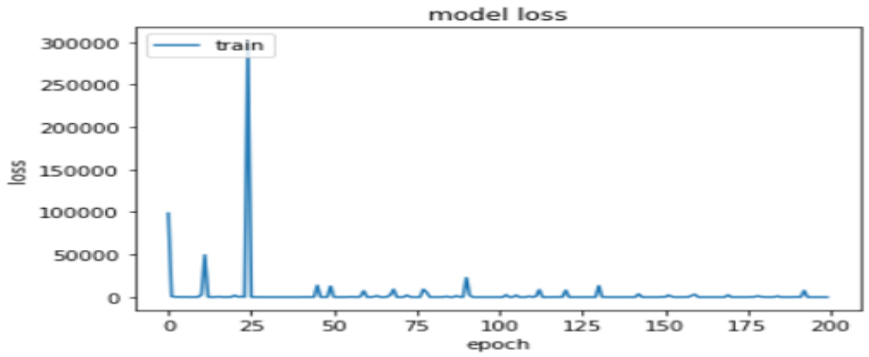

Fig. 11. MSE vs No.of epochs

We define optimizer as 'Adam', loss function as 'Mean squared error' and batch size as 15. When we compiled and constructed a model summary observed that from epoch 1 to 200 the loss function reduced from 74 to 0.8 which is a great result. Refer to Fig 12 for detailed model summary information.

```
Epoch 1/200
2895/2895 [==============================] - 0s 133us/step - loss: 74.0707 - mean_squared_error: 74.0707 - mean_absolute_erro
r: 6.4993
Epoch 2/200
2895/2895 [==============================] - 0s 54us/step - loss: 5.9689 - mean_squared_error: 5.9689 - mean_absolute_error:
1.9043

Epoch 199/200
2895/2895 [==============================] - 0s 55us/step - loss: 0.8570 - mean_squared_error: 0.8570 - mean_absolute_error:
0.6471
Epoch 200/200
2895/2895 [==============================] - 0s 54us/step - loss: 0.8812 - mean_squared_error: 0.8812 - mean_absolute_error:
0.6638

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 40)                720
_________________________________________________________________
dense_2 (Dense)              (None, 20)                820
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 21
=================================================================
Total params: 1,561
Trainable params: 1,561
Non-trainable params: 0
```

Fig. 12. Sequential neural network model summary

3.2.2 LSTM: Neural network like LSTM which are proven to perform very well for time series data. This model poses recurrent neural network architecture that seamlessly adapts better to a multivariate dataset compared to the above discussed machine learning model like section 3.1. As part of data preparation, we derived time attributes like the year, month, date, minutes and hours from the date and time attributes. We plotted a graph refer to Fig 13 with 12 subplots showing the 3 months data for each numeric attribute.

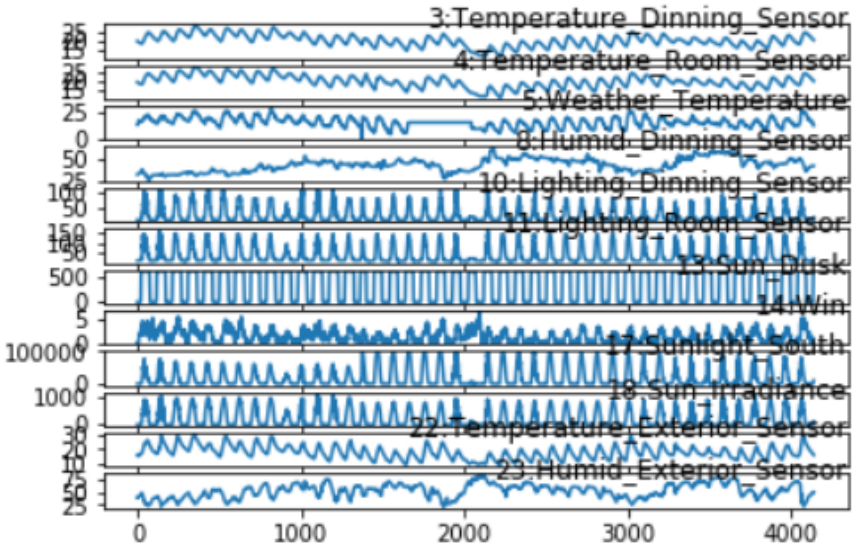

Fig. 13. Timeseries data of numeric attributes

In our project, we want our LSTM model to predict results for every 10 hours. Before converting this time series to supervised learning we normalized data using MinMaxScaler(). We followed this [6] to convert time series into supervised learning and split into train and test data. We constructed the LSTM model using one input, hidden and output layer. On the hidden layer, we defined 50 neurons and the default activation function is 'relu'. We tunned the parameters by trying various combinations of hidden layers, number of neurons and came up with the above optimized model design. Refer to Fig 14, which explains the LSTM model summary that is trained using 'adam' optimizer, 'MSE' loss function, batch size of 15 and 200 epochs. Fig 14 clearly depicts that loss value from 1 to 200 epochs is reduced from 0.13 to 0.013.

We plotted a graph to view how the loss value varied from 1 to 200 epochs for train and test data during the LSTM model training. We observed that refer to Fig 15 train data loss drops below the test data and helps us understand that there is no chance of overfitting the model.

```
Train on 720 samples, validate on 3407 samples
Epoch 1/200
 - 5s - loss: 0.1391 - val_loss: 0.0580
Epoch 2/200
 - 1s - loss: 0.0549 - val_loss: 0.0423
Epoch 3/200
 - 2s - loss: 0.0490 - val_loss: 0.0693
Epoch 4/200
 - 1s - loss: 0.0426 - val_loss: 0.0430
Epoch 5/200
Epoch 195/200
 - 2s - loss: 0.0079 - val_loss: 0.0181
Epoch 196/200
 - 2s - loss: 0.0086 - val_loss: 0.0181
Epoch 197/200
 - 1s - loss: 0.0077 - val_loss: 0.0168
Epoch 198/200
 - 2s - loss: 0.0077 - val_loss: 0.0183
Epoch 199/200
 - 2s - loss: 0.0138 - val_loss: 0.0146
Epoch 200/200
 - 1s - loss: 0.0133 - val_loss: 0.0203
```

Fig. 14. LSTM model summary

Fig. 15. Train and Test Loss from the Multivariate LSTM During Training

4 Data Model Evaluation and Selection

We have analyzed the data, by performing different data preprocessing techniques. We ensured that data is meaningful and clean before fitting to the models. Later, we explained how we constructed each algorithm using various parameters. The below three evaluation parameters we used to evaluate the model's performance:

Pearson Correlation between y_Test and y_Pred: The Pearson correlation evaluates the linear relationship between two continuous variables [7].

Spearman Correlation between y_Test and y_Pred: The Spearman correlation evaluates the monotonic relationship between two continuous or ordinal variables [7].

Mean Squared Error: The Mean Squared Error (MSE) is a measure of how close a fitted line is to data points[8].

We even plotted a graph between the actual and predicted target attribute results for each model. Comparing all these parameters finally, we will state which model suits better for our dataset and helped us achieve our project goal.In this section, we will discuss about the metrics calculated for each algorithm in detailed

4.1 Machine learning models

4.1.1 Linear-based algorithms:

- **a. Linear Regression:** This algorithm is trained on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 2.96, 91% and 91% respectively between the actual result(Y_Test) and predicted results. When we plotted a graph between actual and predicted results we observed still few data instances didn't fit the diagonal line properly refer to Fig 16.

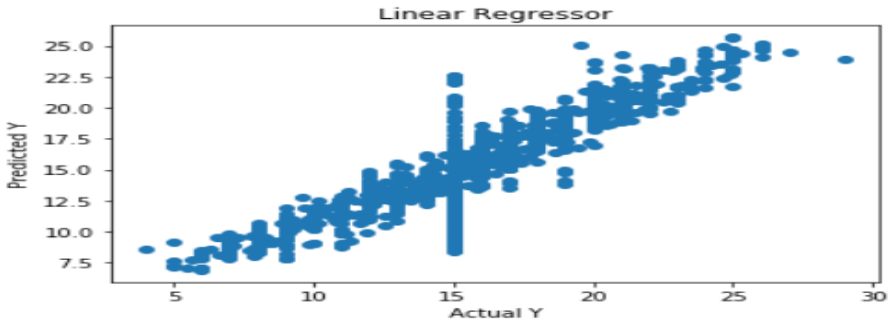

Fig. 16. Actual vs Predicted results of Linear regression model

– **b. Support vector machine:** This algorithm is trained on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 2.44, 93% and 92% respectively between the actual result(Y_Test) and predicted results. Overall, when we compare to Linear this model performance is better. Even, when we plotted a graph between actual and predicted results we observed still few data instances didn't fit the diagonal line properly refer to Fig 17.

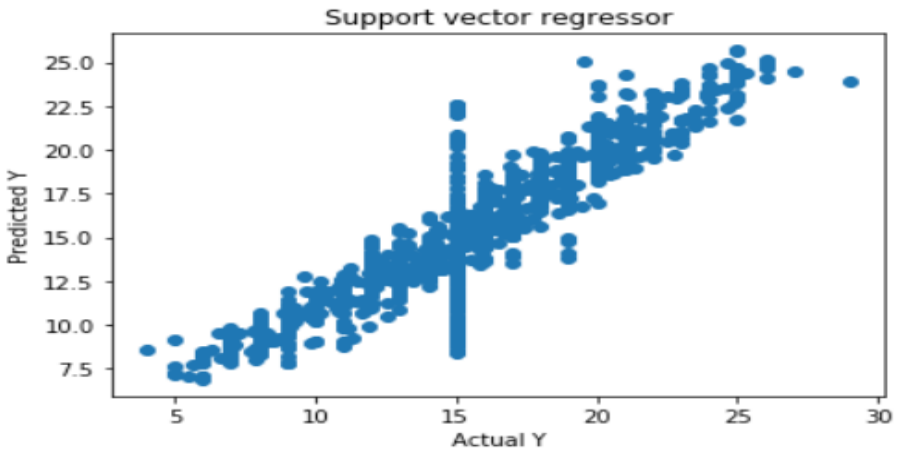

Fig. 17. Actual vs Predicted results of SVM regression model

4.1.2 Distance-based algorithms:

– **a. KNN Regression:** As discussed earlier in section 3.1.2 from the elbow curve we consider k value to be 3. We trained the algorithm on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 1.96, 94% and 94% respectively between the actual result(Y_Test) and predicted results. Compare to algorithms from the linear-based classifiers the KNN model performance is better with less loss and maximum correlation value. When we plotted a graph between actual and predicted results we observed still few data instances didn't fit the diagonal line properly refer to Fig 18.

4.1.3 Tree-based algorithms:

– **a. Decision Tree Regression:** This algorithm is trained on training data (X_Train, Y_Train) and predicted target weather forecast temperature value

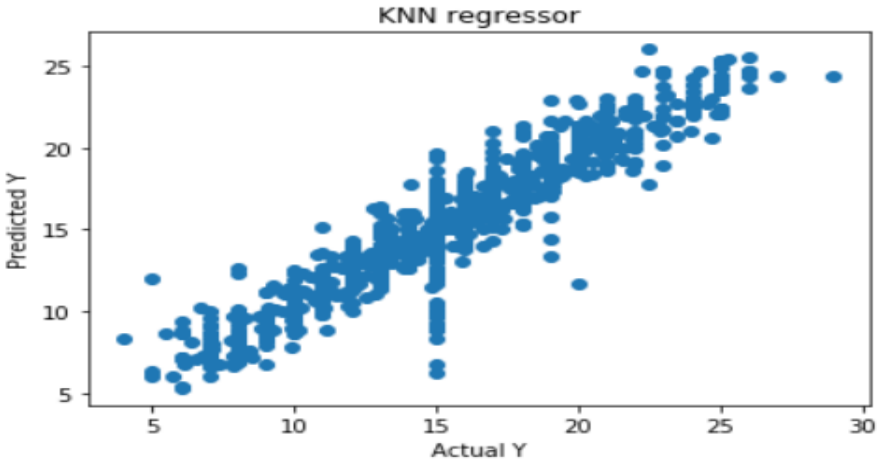

Fig. 18. Actual vs Predicted results of KNN regression model

on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 1.04, 97% and 97% respectively between the actual result(Y_Test) and predicted results. Overall, when we compare to linear,distance this model performance is better. Even, a graph between actual and predicted results we observed the data points fits the line better than linear and distance graphs refer to Fig 19.

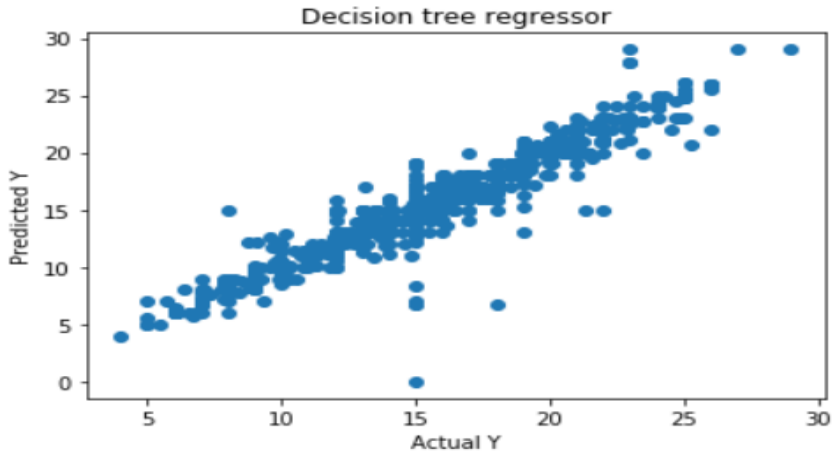

Fig. 19. Actual vs Predicted results of decision tree regression model

4.1.4 Ensemble-based algorithms:

– **a. Boosting Regression:** This algorithm is trained on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 1.15, 96% and 96% respectively between the actual result(Y_Test) and predicted results. When we plotted a graph between actual and predicted results we observed most of the data instances fit the diagonal line properly refer to Fig 20.

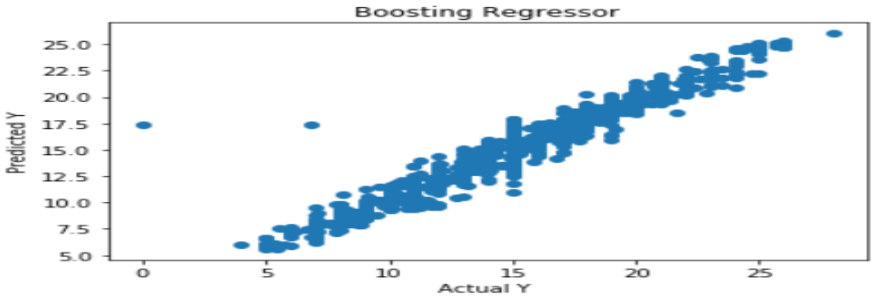

Fig. 20. Actual vs Predicted results of boosting regression model

– **b. Bagging:** This algorithm is trained on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 0.62, 98% and 97% respectively between the actual result(Y_Test) and predicted results. Clearly, this algorithm performs way better than boosting. Overall, when we compare to above all classifiers this model performance is great. Even, when we plotted a graph between actual and predicted results we observed only very few data instances didn't fit the diagonal line properly refer to Fig 21.

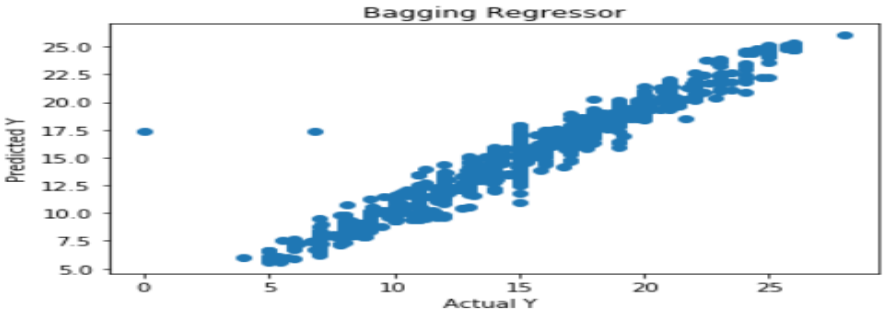

Fig. 21. Actual vs Predicted results of bagging regression model

4.2 Deep learning models

4.2.1 Sequential neural network: We trained the neural network on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 0.9, 97% and 97% respectively between the actual result(Y_Test) and predicted results. We plotted a graph between actual and predicted results we observed only very few data instances didn't fit the diagonal line properly refer to Fig 16.

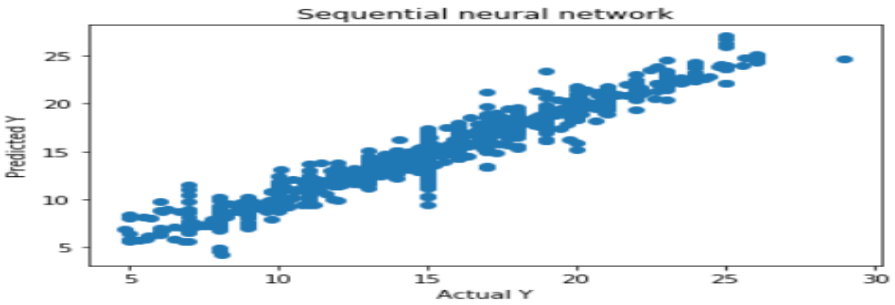

Fig. 22. Actual vs Predicted results of sequential neural network model

4.2.2 LSTM model: We trained the neural network on training data (X_Train, Y_Train) and predicted target weather forecast temperature value on test data(X_Test). The calculated MSE, Pearson, and Spearman values are 0.025, 99% and 99% respectively between the actual result(Y_Test) and predicted results. The graph plotted between actual and predicted results refer to Fig 23 perfectly fits the diagonal line. We can clearly say that, this model performed exceptionally well with minimum loss of 0.025 and maximum correlation about 99

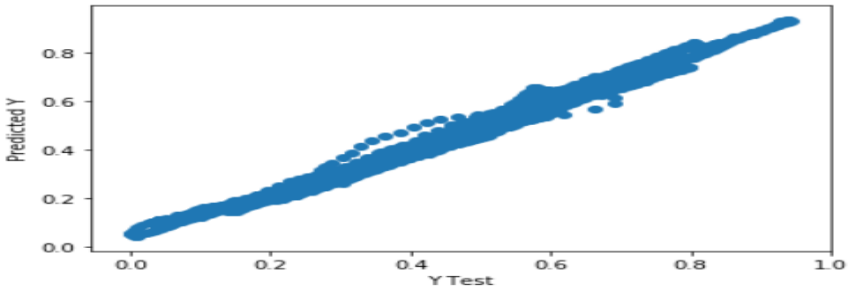

Fig. 23. Actual vs Predicted results of LSTM model

4.3 Comparison of model performances:

Model	Pearson Correlation	Spearman Correlation	Mean Squared Error
Linear Regression	0.916	0.919	2.96
Support Vector Regression	0.93	0.92	2.44
KNN Regression	0.942	0.948	1.96
Decision Tree Regression	0.960	0.97	1.04
Gradient Boosting Regression	0.96	0.96	1.15
Bagging Regression	0.982	0.978	0.62
Sequential Deep Learning	0.921	0.926	0.9
LSTM Deep Learning	0.991	0.993	0.025

Fig. 24. Summary of all algorithms metrics

The aim of this project is to select a model that gives accurate predictions with minimum loss and maximum correlation value. We summarised all the metrics into fig 24 and clearly state that the bagging model from machine learning and the LSTM model from deep learning models performed exceptionally well. This dataset is a time series regression problem we select the LSTM model as the best one among all of them since it's loss value is 0.025 which is very minute and maximum accuracy of 99

5 Conclusion and Future work

Comparing all the results of the algorithms, we conclude stating that the LSTM model suits best for our dataset. This project goal was to build a model that predicts weather forecast temperature accurately with minimum mean squared error and maximum correlation value. The LSTM deep learning model accuracy is 99% and loss value is very minimum about 0.025. This model's recurrent neural network architecture remembers the previous patterns for a long duration. It also handles the lags of unknown data between the events in time series[10].

As part of the future we can use anomaly detection algorithms like Elliptic Envelope, Local Outlier Factor, or Isolation Forests in order to detect which days of the three months exhibited anomalous behavior.Since the modeling considers various indoor attributes and there are chances of less occupancy of the smart home during weekdays than on weekends, explore data prediction based on weekends and weekdays.

Acknowledgment

We want to thank Professor Dr.Olubisi Runsewe, University of Ottawa, for her help and valuable guidance throughout this project.

References

1. https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f. [Last accessed: 16-04-2020]
2. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Last accessed: 16-04-2020]
3. https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d.[Last accessed on 16-04-2020]
4. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models.[Last accessed on 16-04-2020].
5. https://www.analyticsvidhya.com/blog/2016/04/complete-tutorialtree-based-modeling-scratch-in-python/. [Last accessed on 16-04-2020].
6. https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/[Last accessed on 17-04-2020].
7. https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods/[Last accessed on 17-04-2020].
8. https://www.vernier.com/til/1014.[Last accessed on 17-04-2020].
9. https://archive.ics.uci.edu/ml/datasets/SML2010.[Last accessed on 18-04-2020].
10. https://en.wikipedia.org/wiki/Long_short-term_memory.[Last accessed on 19-04-2020].
11. https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/[Last accessed on 19-04-2020].
12. https://keras.io/[Last accessed on 19-04-2020]. .