Literature Review and Exploratory Data Analysis of Predicting

Name: Peter Yu, 501137007

Date: June 20th, 2022

Supervised by: Dr. Alan Fung

Background

The purpose of this study is to explore some of the models used for forecasting household temperatures using local weather station data, as well as examine the types of features that could potentially help with the related task of time series forecasting future household temperatures, as well as the adjacent task of feature engineering. Since there is no formal structuring of the literature review, the plan is to go through several recent academic papers, and summarize their methodologies, results, and findings.

Literature Review

Modelling residential house electricity demand profile and analysis of peaksaver program using ANN: Case study for Toronto, Canada - 2016

Energy management has been a growing concern of the 21st century given the recent national agreements where various countries have agreed upon aiming for greener solutions as well as the increasing energy prices around the world. This is a two-sided problem as on the demand management side, electricity cannot be easily stored in large quantities, thus electricity companies need to estimate the number of electricity generators needed to meet customer power demands at varying times during the day. Since there is varying demand in electricity, utility companies cannot simply just build more power generators, as sudden drops of electricity

demand would result in reduced revenue. Instead, one could investigate machine learning to try to predict energy demand. Essentially Poulad, et al., created an artificial neural network trained on Toronto weather data labels including weather condition, date, temperature, humidity, wind direction, wind speed, visibility, and the average power demand as the ground truth value. They used various data from 2008-2011 along with several customers to average their energy demand, and then they evaluated the artificial neural network performance on various days in the summer, and winter, where the actual energy demand was compared with the predicted throughout the day, and its error was evaluated with Pearson's coefficient. Having done that, they then evaluated their trained model by backtesting during certain periods, and calculating the cost of electricity, to see whether their model would save customers money. Their findings were that the ANN was able to predict electricity demand in summer as well as winter very accurately (R^2 was roughly around 0.99) and that energy savings were easily achieved. One thing to note is since different models use different time periods for training and testing, for simpler generalizability, it might be worth using the same dataset for comparing between other models, as well as looking into other error metrics rather than R^2, as Pearson's coefficient loses value with increases in the number of features.

Predicting Indoor Temperature From Smart Thermostat And Weather Forecast Data

In a similar paper, weather station data was then coupled with thermostat data, to try to predict indoor household temperatures. By predicting and forecasting indoor household temperatures, smart thermostats can optimize residential HVAC systems and make buildings more "comfortable" for its tenants. Essentially Yu et al., coupled data from 16 Ecobee thermostats with solar radiation data (namely average indoor temperature, heating/cooling setpoints, humidity, fan runtime, outdoor temperature, and solar radiation) from 2015 and 2016, and

trained both a generalized linear regression neural network (GRNN) and artificial neural network (ANN) individually on an 80/20 data split. They used mean squared error as an error metric and found that the MSEs were 0.79 and 11.52 for the GRNN and ANN respectively. Their findings were that by adding additional features outside of thermostat data, the model performance was improved upon.

Forecasting The Air Temperature at a Weather Station Using Deep Neural Networks

In a similar topic, apart from the papers published alongside Dr. Fung, was a paper on forecasting air temperature at a weather station using deep neural networks. This paper was chosen as it seemed like the current work being conducted (i.e., using a weather station as a mock representation of a house, and using its weather data to forecast for future temperatures). The author proposed three different neural network architectures from a baseline multi-layer perceptron neural network (MLP) to a Long Short Term Memory Network (LSTM), to finally a Convolutional Neural Network with LSTM (CNN + LSTM) model. By using average wind speed, precipitation, snowfall, snow depth, average temperature, maximum temperature, and minimum temperature from JFK International Airport with data from Jan 1st 2009 to Jan 1st 2019, three neural networks were evaluated for their RMSE and MAPE scores. They found that with the CNN+LSTM model, a 97.42% MAPE was obtained, which was nearly 2% better than the LSTM model and 8% better than the MLP model when forecasting one day ahead. The models suffer when looking forecasting ten days ahead, where the CNN+LSTM model drops to a MAPE of 71%. For future studies, the authors proposed adding more weather variables.

Exploratory Data Analysis

As this is an ongoing research project, currently the data on hand is from four different weather stations, including stations in Toronto city, Toronto City Center, Toronto International Airport, and Buttonville. This results in three neighbouring weather stations, with Toronto City being the weather station situated in between the other three. When loading our data in, we get the following dataset:

Latitude (y)	Station Name	Climate ID	Date/Time (LST)	Year	Month	Day	Time (LST)	Temp (°C)	...	Wind Spd Flag	Visibility (km)	Visibility Flag	Stn Press (kPa)	Stn Press Flag	Hmdx	Hmdx Flag	Wind Chill	Wind Chill Flag	Weather
43.67	TORONTO CITY	6158355	2016-01-01 00:00	2016	1	1	00:00	0.8	...	M	NaN	NaN	100.38	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2016-01-01 01:00	2016	1	1	01:00	0.6	...	M	NaN	NaN	100.35	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2016-01-01 02:00	2016	1	1	02:00	0.5	...	M	NaN	NaN	100.30	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2016-01-01 03:00	2016	1	1	03:00	0.4	...	M	NaN	NaN	100.27	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2016-01-01 04:00	2016	1	1	04:00	0.4	...	M	NaN	NaN	100.27	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
43.67	TORONTO CITY	6158355	2021-12-31 19:00	2021	12	31	19:00	6.4	...	NaN	NaN	NaN	99.59	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2021-12-31 20:00	2021	12	31	20:00	5.1	...	NaN	NaN	NaN	99.60	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2021-12-31 21:00	2021	12	31	21:00	4.7	...	NaN	NaN	NaN	99.51	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2021-12-31 22:00	2021	12	31	22:00	4.5	...	NaN	NaN	NaN	99.45	NaN	NaN	NaN	NaN	NaN	NaN
43.67	TORONTO CITY	6158355	2021-12-31 23:00	2021	12	31	23:00	4.6	...	NaN	NaN	NaN	99.41	NaN	NaN	NaN	NaN	NaN	NaN

with the following data types:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 56232 entries, 0 to 743
Data columns (total 30 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Longitude (x)         56232 non-null  float64
 1   Latitude (y)          56232 non-null  float64
 2   Station Name          56232 non-null  object
 3   Climate ID            56232 non-null  int64
 4   Date/Time (LST)       56232 non-null  object
 5   Year                  56232 non-null  int64
 6   Month                 56232 non-null  int64
 7   Day                   56232 non-null  int64
 8   Time (LST)            56232 non-null  object
 9   Temp (°C)             55619 non-null  float64
 10  Temp Flag             9 non-null      object
 11  Dew Point Temp (°C)   55621 non-null  float64
 12  Dew Point Temp Flag   8 non-null      object
 13  Rel Hum (%)           55621 non-null  float64
 14  Rel Hum Flag          8 non-null      object
 15  Precip. Amount (mm)   55076 non-null  float64
 16  Precip. Amount Flag   553 non-null    object
 17  Wind Dir (10s deg)    0 non-null      float64
 18  Wind Dir Flag         29825 non-null  object
 19  Wind Spd (km/h)       0 non-null      float64
 20  Wind Spd Flag         29825 non-null  object
 21  Visibility (km)       0 non-null      float64
 22  Visibility Flag       0 non-null      float64
 23  Stn Press (kPa)       55596 non-null  float64
 24  Stn Press Flag        33 non-null     object
 25  Hmdx                  9235 non-null   float64
 26  Hmdx Flag             0 non-null      float64
 27  Wind Chill            0 non-null      float64
 28  Wind Chill Flag       0 non-null      float64
 29  Weather               0 non-null      float64
dtypes: float64(16), int64(4), object(10)
```

When looking at the above data, my two concerns are having using too many correlated features for forecasting (which could potentially result in a curse of dimensionality scenario) as well as dealing with the NaN values, which seemed to occur quite commonly. This was the same case for the other three dataframes which contained the other weather station data. Using Pandas API, the NaN values were quickly calculated, and the below image showed that there were quite a large number of NaN values for certain features, which resulted in my decision to drop any of the features with >50,000 NaN row values (as my dataset had only ~55,000 rows).

```
40
41 print(toronto_city_df.isna().sum())
42 print(toronto_city_center_df.isna().sum())
43 print(toronto_intl_a_df.isna().sum())
44 print(buttonville_df.isna().sum())
```

```
Date                  0
weather_label_tc    56232
wind_spd_tc         56232
dew_point_tc          611
actual_temp           613
wind_dir_tc         56232
dtype: int64
Date                    0
weather_label_tcc    52852
wind_spd_tcc          5632
dew_point_tcc         5818
temp_tcc              5718
wind_dir_tcc          8684
dtype: int64
Date                    0
weather_label_tia    37105
wind_spd_tia          5554
dew_point_tia         5556
temp_tia              5555
wind_dir_tia          5577
dtype: int64
Date                  0
weather_label_b    61368
wind_spd_b          6024
dew_point_b         5962
temp_b              5963
wind_dir_b          6089
dtype: int64
```

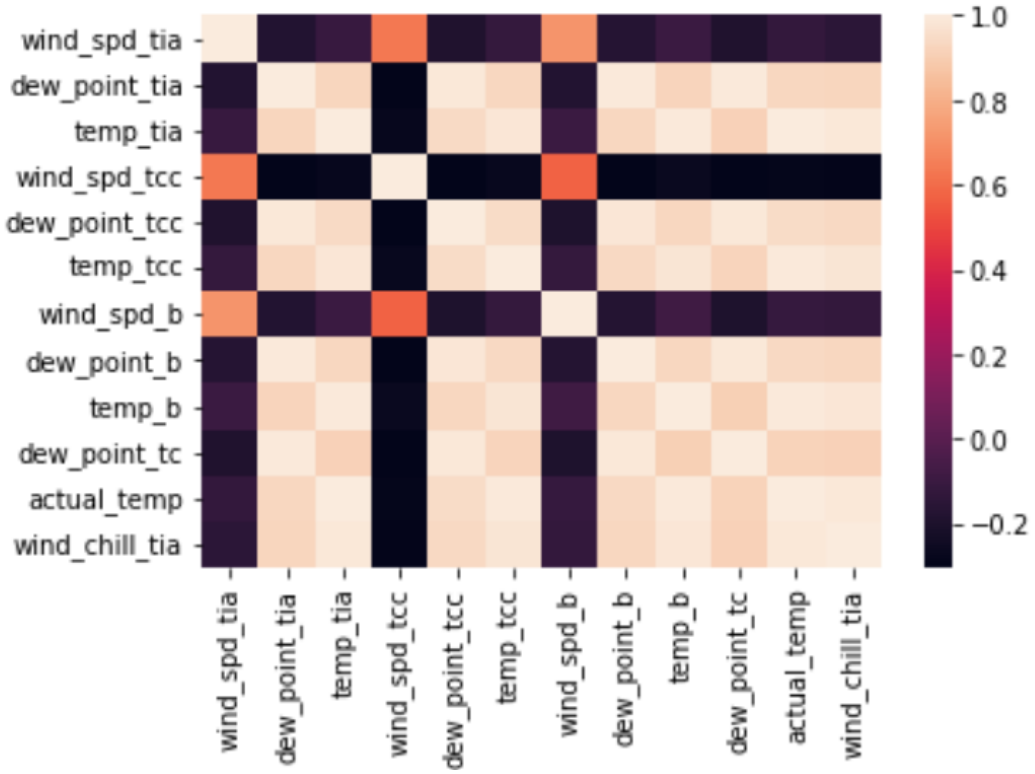
In addition to that, I also dropped a number of the columns present in the initial dataframe based on my *a priori* knowledge of which features were very correlated with one another, while keeping in mind that for future analysis, some of these features may be included again. To give context to the EDA process, I also converted the wind direction (in 10s degree) to compass rose bearings, and converted them into dummy columns, as well as added wind chill as a feature to my dataset. Humidex was omitted for now as the following figure showed that the mean was 8.7, while the humidex was only really useful for Humidex values ranging from 20-45+.

```
Out[47]: count    55488.000000
         mean         8.723461
         std         14.290469
         min        -31.299221
         25%         -2.411590
         50%          6.757091
         75%         20.928156
         max         45.580289
         dtype: float64
```

By using the .describe() method, my data examination process could be expedited as shown below:

	wind_spd_tia	dew_point_tia	temp_tia	wind_spd_tcc	dew_point_tcc	temp_tcc	wind_spd_b	dew_point_b	temp_b	dew_point_tc
count	55488.000000	55488.000000	55488.000000	55488.000000	55488.000000	55488.000000	55488.000000	55488.000000	55488.000000	55488.000000
mean	16.650195	2.977139	9.010164	17.225022	4.067045	8.984467	13.452134	2.760391	8.359835	2.805511
std	9.210448	10.458427	11.039394	10.214484	10.262651	9.927586	8.030069	10.583003	11.198902	9.965201
min	0.000000	-31.600000	-26.000000	0.000000	-30.600000	-24.200000	0.000000	-33.000000	-26.600000	-31.900000
25%	10.000000	-4.700000	0.600000	9.000000	-3.200000	1.500000	7.000000	-4.900000	-0.100000	-4.400000
50%	15.000000	2.400000	8.100000	15.000000	3.400000	7.900000	12.000000	2.200000	7.500000	2.300000
75%	22.000000	11.800000	18.600000	22.000000	12.800000	17.900000	18.000000	11.700000	17.800000	11.100000
max	80.000000	25.100000	35.200000	76.000000	25.200000	34.500000	60.000000	25.200000	35.800000	25.100000

which showed that given my *a priori* knowledge, there weren't any extreme data values in my dataset (i.e., temperature of 50 Celcius which would indicate sensor failure). When generating a heatmap, we then see some good news, where most of the features are not too correlated with one another, which is good because that means less feature engineering is required later on.

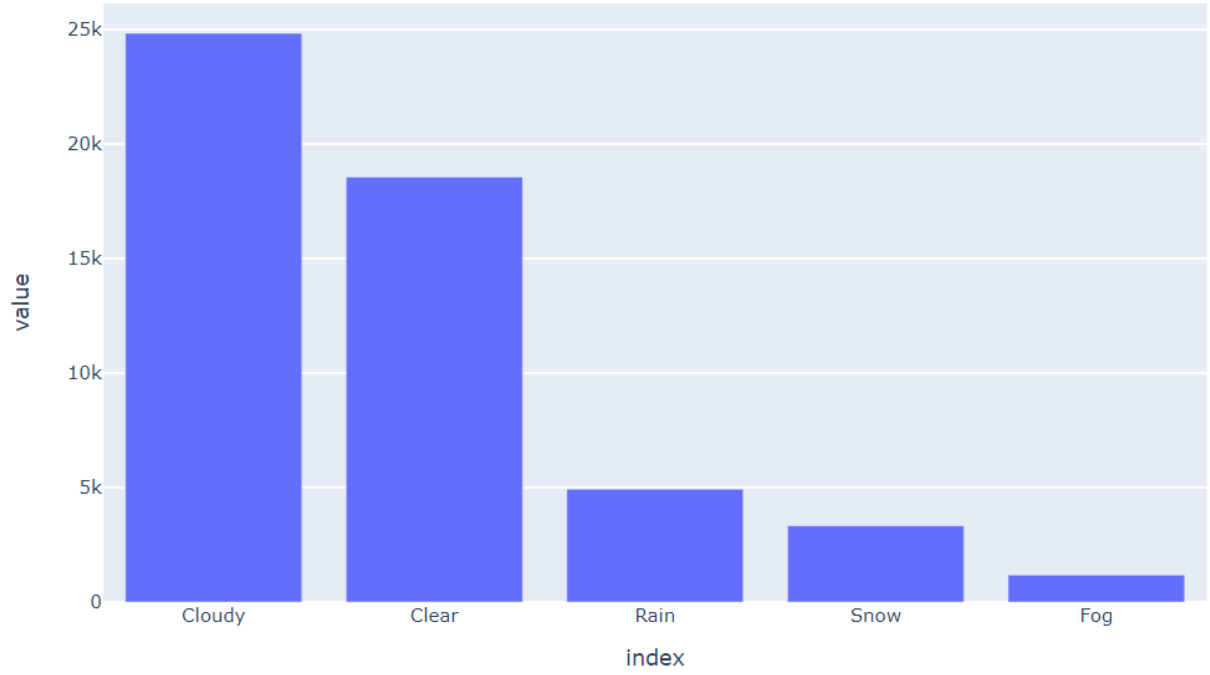


```
1 combined_df.corr()
```

	wind_spd_tia	dew_point_tia	temp_tia	wind_spd_tcc	dew_point_tcc	temp_tcc	wind_spd_b	dew_point_b	temp_b	dew_point_tc	actual_temp
wind_spd_tia	1.000000	-0.185066	-0.112867	0.631089	-0.196202	-0.124794	0.711329	-0.173880	-0.103724	-0.194642	-0.126806
dew_point_tia	-0.185066	1.000000	0.927548	-0.301076	0.987243	0.932062	-0.184205	0.990370	0.920605	0.991858	0.931435
temp_tia	-0.112867	0.927548	1.000000	-0.278862	0.939656	0.983415	-0.100460	0.930860	0.993943	0.911935	0.995716
wind_spd_tcc	0.631089	-0.301076	-0.278862	1.000000	-0.302027	-0.274229	0.567922	-0.301242	-0.268592	-0.303519	-0.288640
dew_point_tcc	-0.196202	0.987243	0.939656	-0.302027	1.000000	0.947516	-0.198491	0.982410	0.932087	0.988784	0.944777
temp_tcc	-0.124794	0.932062	0.983415	-0.274229	0.947516	1.000000	-0.120224	0.936065	0.976669	0.920203	0.990552
wind_spd_b	0.711329	-0.184205	-0.100460	0.567922	-0.198491	-0.120224	1.000000	-0.179207	-0.084782	-0.197577	-0.118119
dew_point_b	-0.173880	0.990370	0.930860	-0.301242	0.982410	0.936065	-0.179207	1.000000	0.929043	0.986256	0.935306
temp_b	-0.103724	0.920605	0.993943	-0.268592	0.932087	0.976669	-0.084782	0.929043	1.000000	0.903533	0.990825
dew_point_tc	-0.194642	0.991858	0.911935	-0.303519	0.988784	0.920203	-0.197577	0.986256	0.903533	1.000000	0.917029
actual_temp	-0.126806	0.931435	0.995716	-0.288640	0.944777	0.990552	-0.118119	0.935306	0.990825	0.917029	1.000000

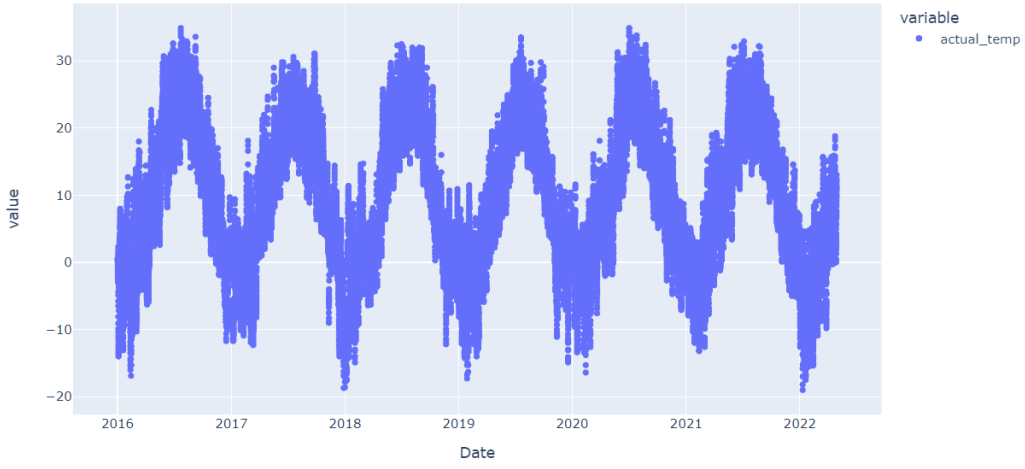
The outliers were then scrubbed by removing values that were outside the 1.5*IQR+Q3 and Q1-1.5*IQR range. Having cleaned up my final dataframe, the final steps were adding lags to the features as the plan was to forecast for the next hourly periods using 1-hour lagged data, as well as converting the weather labels to something manageable, as there were 80 labels for weather. This procedure was done by converting labels like “Heavy rain”, or “Light rain” into one generalizable label titled “rain”, and simplifying many of the labels. In the end, the distribution

of labels was:



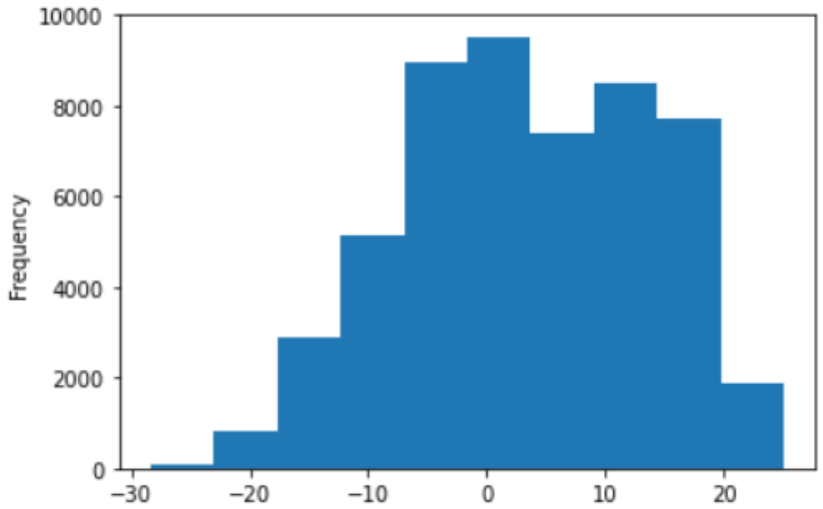
As a final check, some of the dataframe values were also plotted for to confirm that the data did not contain any sizeable outliers.

Plot of the ground truth temperature



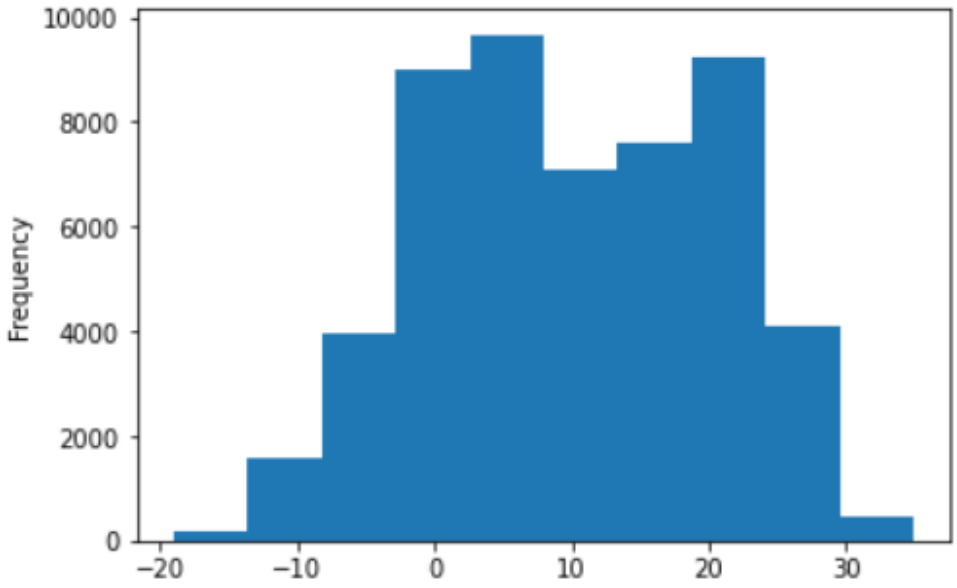
Histogram of the dew point distribution, which should be very similar to the temperature as they are very correlated with one another:

```
In [79]:  1 final_df['dew_point_tia_1'].plot.hist()
          2 plt.show()
```



Histogram of the actual temperature distribution which aligns with the dew point being correlated.

```
80]:  1 final_df['actual_temp'].plot.hist()
      2 plt.show()
```



References

1. Poulad, M. E., Department, M. and I. E., Fung, A. S., He, L., Colpan, C. O., & Engineering, D. of M. (2016, July 25). *Modelling Residential House Electricity Demand Profile and analysis of peaksaver program using ANN: Case study for Toronto, Canada*. International Journal of Global Warming. Retrieved June 20, 2022, from https://www.inderscienceonline.com/doi/abs/10.1504/IJGW.2016.077911
2. University, D. Y. R., Yu, D., University, R., University, A. A. R., Abhari, A., University, A. S. F. R., Fung, A. S., University, K. R. R., Raahemifar, K., University, F. M. R., Mohammadi, F., University, S., & Metrics, O. M. V. A. (2018, April 1). *Predicting indoor temperature from Smart Thermostat and weather forecast data: Proceedings of the communications and networking symposium*. ACM Conferences. Retrieved June 20, 2022, from https://dl.acm.org/doi/abs/10.5555/3213200.3213209
3. Roy, D. S. (2020, December 7). *Forecasting the air temperature at a weather station using Deep Neural Networks*. Procedia Computer Science. Retrieved June 20, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050920323784