Weather in Szeged 2006-2016

Regression based analysis for predicting apparent temperatures based on Hourly/daily weather summary with temperature, pressure, wind speed and more parameters

An assignment submitted in fulfillment of the requirements of IBM Introduction to Machine Learning Specialization

By

Minhaj Ahmed Ansari

1. Introduction

1.1. The target of this machine learning exercise is to identify the best model to predict the apparent temperature using the input variables available in the data set.

1.2. Apparent temperature is the temperature equivalent perceived by humans, caused by the combined effects of air temperature, relative humidity, and wind speed. We will be focusing more on the prediction rather the interpretation of the results. This is because we already know the major factors which impact the Apparent temperature.

2. Brief description of the data set and a summary of its attributes

2.1. The Dataset which I have chosen for this assignment is publicly available on Kaggle titled '**Weather in Szeged 2006-2016'.** The data contains Hourly/daily summary with temperature, pressure, wind speed and more for the city name Szeged in Hungary.

2.2. Data available in the hourly response:

- Time – Time of Observation -
- Summary – Brief summary of weather
- PrecipType – Type of precipitation
- Temperature
- apparentTemperature – Apparent temperature is the temperature equivalent perceived by humans, caused by the combined effects of air temperature, relative humidity, and wind speed.
- Humidity
- windSpeed
- windBearing
- visibility
- CloudCover
- Pressure

2.3. The total number of attributes are 12 and the total number of observations are 96453.

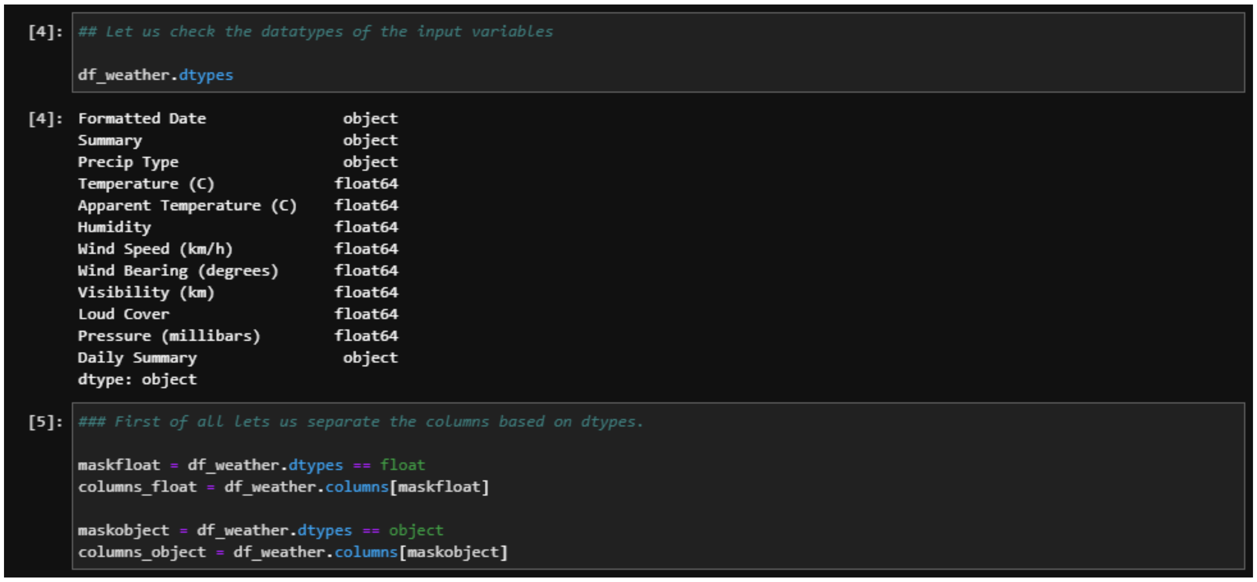
3. Data Cleaning and Wrangling

3.1. Plan

- To check the missing values in the dataset and suitably replace them or drop the rows.
- Identify which variables are constant or very low variance, thereby discarding them so that the dataset could be optimized.

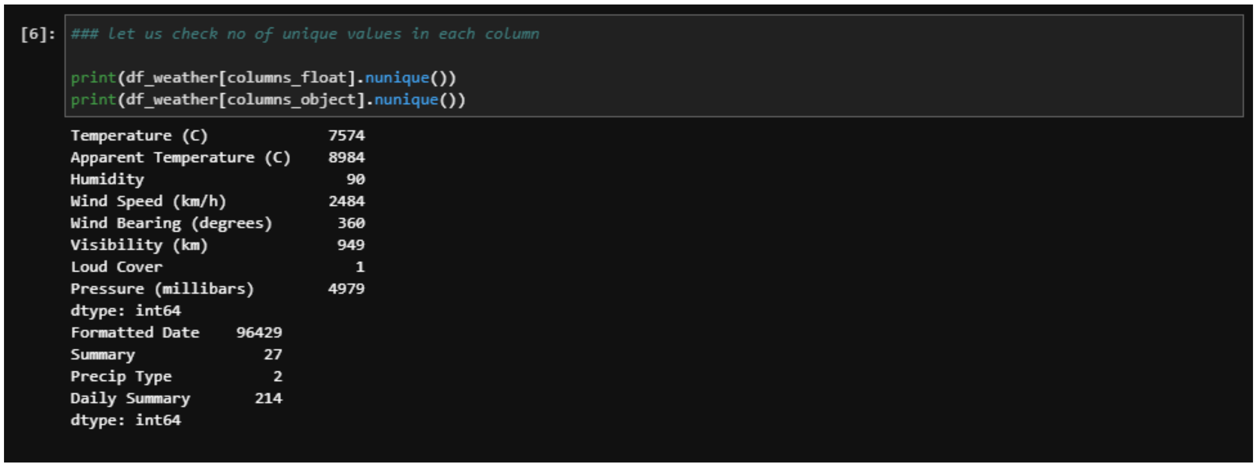
3.2. Actions

- Creating the masks for float and object datatype columns, which could be utilized later for EDA.



```
[4]: ## Let us check the datatypes of the input variables

     df_weather.dtypes

[4]: Formatted Date               object
     Summary                      object
     Precip Type                  object
     Temperature (C)             float64
     Apparent Temperature (C)    float64
     Humidity                    float64
     Wind Speed (km/h)           float64
     Wind Bearing (degrees)      float64
     Visibility (km)             float64
     Loud Cover                  float64
     Pressure (millibars)        float64
     Daily Summary                object
     dtype: object

[5]: ### First of all lets us separate the columns based on dtypes.

     maskfloat = df_weather.dtypes == float
     columns_float = df_weather.columns[maskfloat]

     maskobject = df_weather.dtypes == object
     columns_object = df_weather.columns[maskobject]
```

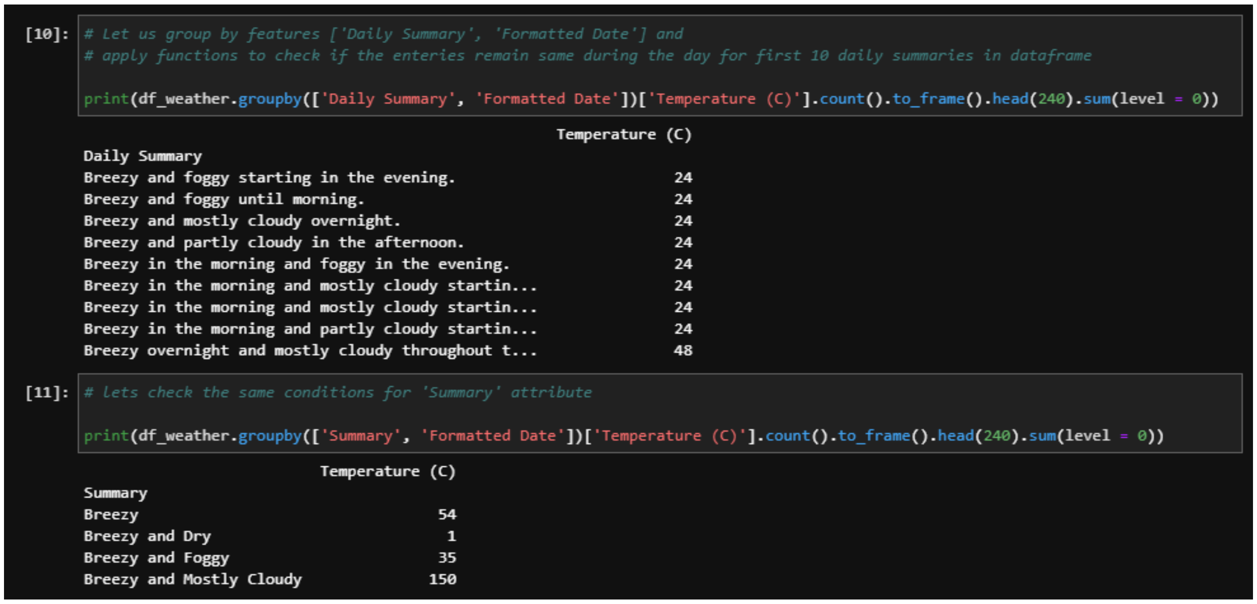
- Checking No of unique values in each column. Dropping columns which contain all unique values and constant values. These are 'Formatted date' and 'loud cover' variables.



```
[6]: ### Let us check no of unique values in each column

     print(df_weather[columns_float].nunique())
     print(df_weather[columns_object].nunique())

     Temperature (C)             7574
     Apparent Temperature (C)    8984
     Humidity                      90
     Wind Speed (km/h)           2484
     Wind Bearing (degrees)       360
     Visibility (km)              949
     Loud Cover                     1
     Pressure (millibars)        4979
     dtype: int64
     Formatted Date    96429
     Summary              27
     Precip Type           2
     Daily Summary       214
     dtype: int64
```

- Next I checked for columns with missing values, this led to identification of 'Precip type' variable which had around 517 missing values. I dropped these rows as they are very miniscule part of overall dataset.
- I identified that the categorical attribute called 'Daily Summary' remains almost constant during the entire day (We have 24 observations for each day in the

dataset). It can be safely assumed that this variable possibly cannot contribute any meaningful information to the modeling. Therefore, I dropped this column too.



```
[10]: # Let us group by features ['Daily Summary', 'Formatted Date'] and
      # apply functions to check if the enteries remain same during the day for first 10 daily summaries in dataframe

      print(df_weather.groupby(['Daily Summary', 'Formatted Date'])['Temperature (C)'].count().to_frame().head(240).sum(level = 0))

                                                    Temperature (C)
      Daily Summary
      Breezy and foggy starting in the evening.                   24
      Breezy and foggy until morning.                             24
      Breezy and mostly cloudy overnight.                         24
      Breezy and partly cloudy in the afternoon.                  24
      Breezy in the morning and foggy in the evening.             24
      Breezy in the morning and mostly cloudy startin...          24
      Breezy in the morning and mostly cloudy startin...          24
      Breezy in the morning and partly cloudy startin...          24
      Breezy overnight and mostly cloudy throughout t...          48

[11]: # lets check the same conditions for 'Summary' attribute

      print(df_weather.groupby(['Summary', 'Formatted Date'])['Temperature (C)'].count().to_frame().head(240).sum(level = 0))

                                Temperature (C)
      Summary
      Breezy                                 54
      Breezy and Dry                          1
      Breezy and Foggy                       35
      Breezy and Mostly Cloudy              150
```

- Another thing that I check for was No. of Zero values in each column. There were many zero values, but all are very much plausible except for pressure which cannot be 0, otherwise we would have vacuum in Szeged. This may be due to measurement error. So, I filled these rows from previous observation.

3.3. Summary

- Since we are not doing time-series analysis, I have dropped the Formatted date column from our data
- loud cover (I think it is typo for Cloud Cover) attribute is a constant. So we can drop it.
- We tested categorical variables such as Precip type, Daily Summary and Summary attributes for information they contribute to the dataset. Preliminary analysis suggests that we can safely drop Daily Summary data as it remains constant over a day.
- We also checked the null values and 0 values. We dropped observations with null values in precip type and treated 0 values in pressure column, because if we don't do that, it practically means szeged had perfect vacuum atmosphere for those many time periods!!

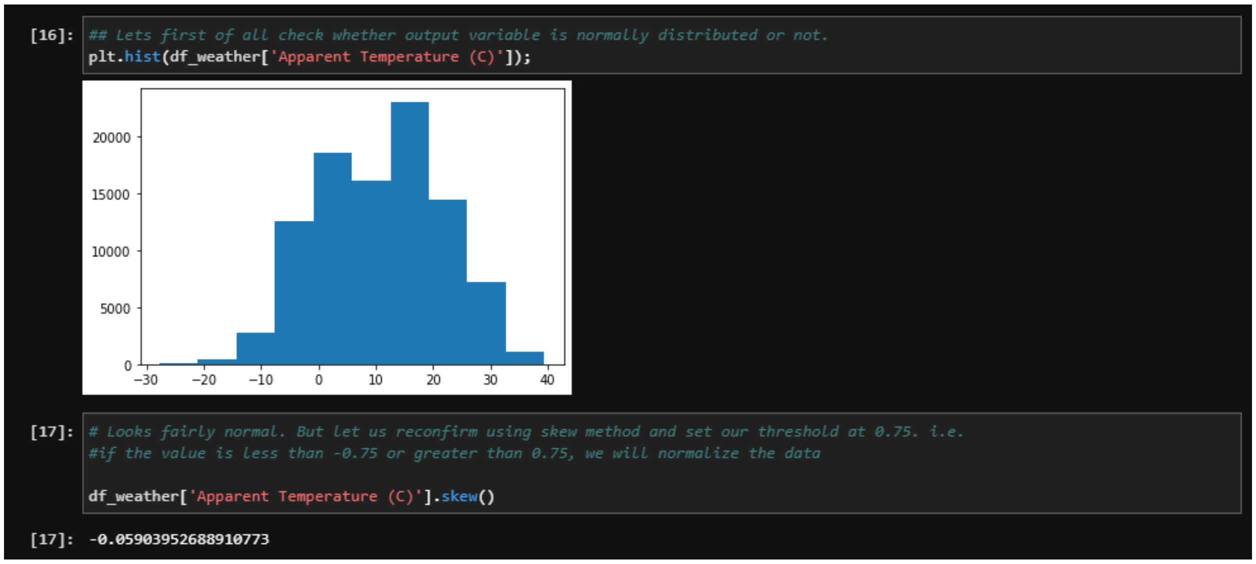
4. Exploratory Data Analysis

4.1. Plan

- To check the skewness of the target variable i.e. Apparent Temperature and treat it in case it is present.
- Forming hypothesis statements for selection Categorical Statement. Then One-hot encoding them for generating features
- Checking the skewness of the input numerical variables and treat them in case it is present.
- Checking if there exists any collinearity between the input variables and drop duplicate columns in case it exits.

4.2. Actions

- I utilized hist plot as well as skew function of pandas to identify if there exists substantial skewness in the target data (less than -0.75 or greater than 0.75). However, investigation suggested that the target variable is almost normally distributed, thus doesn't require any transformation.

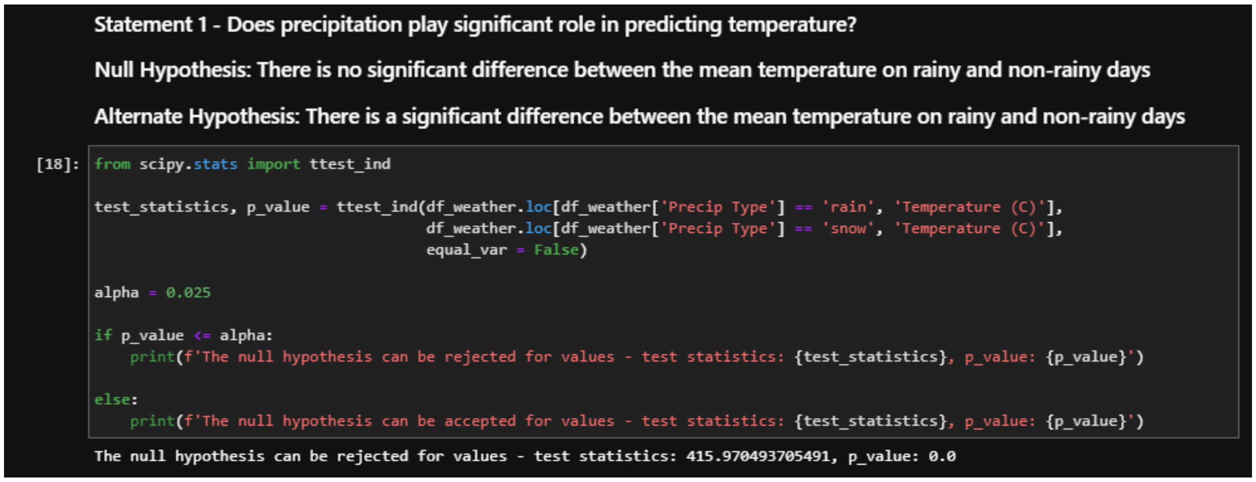


```
[16]: ## Lets first of all check whether output variable is normally distributed or not.
      plt.hist(df_weather['Apparent Temperature (C)']);
```

```
[17]: # Looks fairly normal. But let us reconfirm using skew method and set our threshold at 0.75. i.e.
      #if the value is less than -0.75 or greater than 0.75, we will normalize the data

      df_weather['Apparent Temperature (C)'].skew()

[17]: -0.05903952688910773
```

- For selecting Categorical variables, I tested two hypotheses.
  - ✓ Does precipitation play significant role in predicting temperature? – For 'precip type' variable.
  - ✓ Does mean temperature varies significantly for different summaries? – For 'Summary' variable.

- For first statement a standard p_value z-test suggested that there is significant difference in means for each categorical value under ‘precip type’ variable. Thus we keep this variable.



Statement 1 - Does precipitation play significant role in predicting temperature?

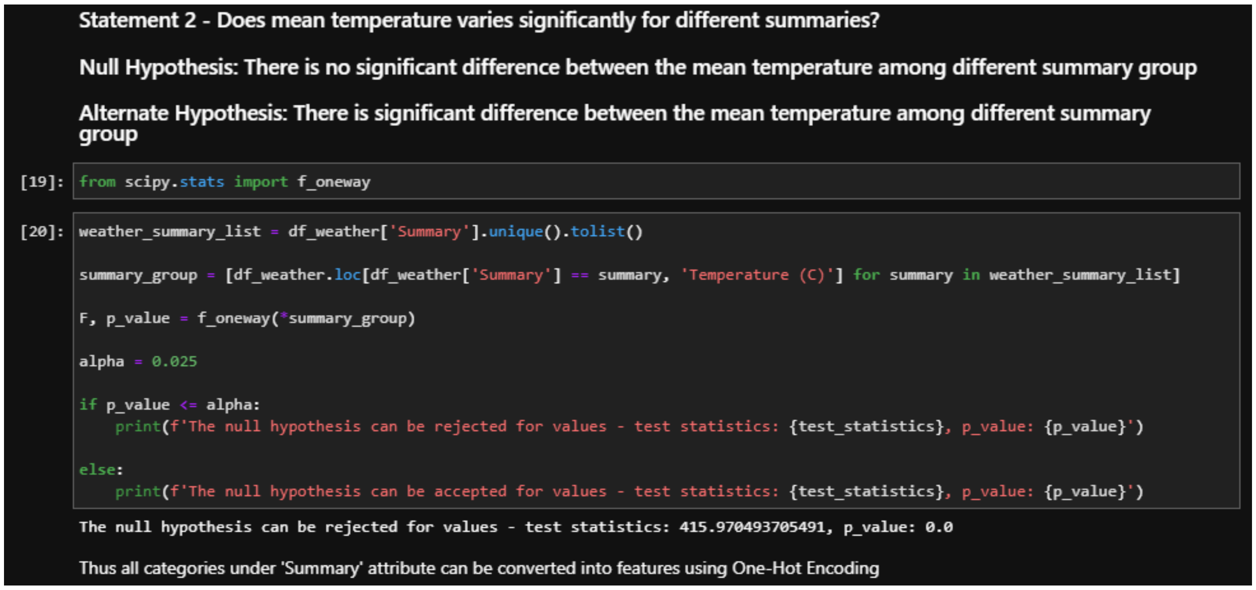
Null Hypothesis: There is no significant difference between the mean temperature on rainy and non-rainy days

Alternate Hypothesis: There is a significant difference between the mean temperature on rainy and non-rainy days

```
[18]: from scipy.stats import ttest_ind

test_statistics, p_value = ttest_ind(df_weather.loc[df_weather['Precip Type'] == 'rain', 'Temperature (C)'],
                                     df_weather.loc[df_weather['Precip Type'] == 'snow', 'Temperature (C)'],
                                     equal_var = False)

alpha = 0.025

if p_value <= alpha:
    print(f'The null hypothesis can be rejected for values - test statistics: {test_statistics}, p_value: {p_value}')

else:
    print(f'The null hypothesis can be accepted for values - test statistics: {test_statistics}, p_value: {p_value}')
```

```
The null hypothesis can be rejected for values - test statistics: 415.970493705491, p_value: 0.0
```

- For second test, a one-way ANOVA test was formulated, which again suggested significant mean different between each categorical values under ‘Summary’ variable.



Statement 2 - Does mean temperature varies significantly for different summaries?

Null Hypothesis: There is no significant difference between the mean temperature among different summary group

Alternate Hypothesis: There is significant difference between the mean temperature among different summary group

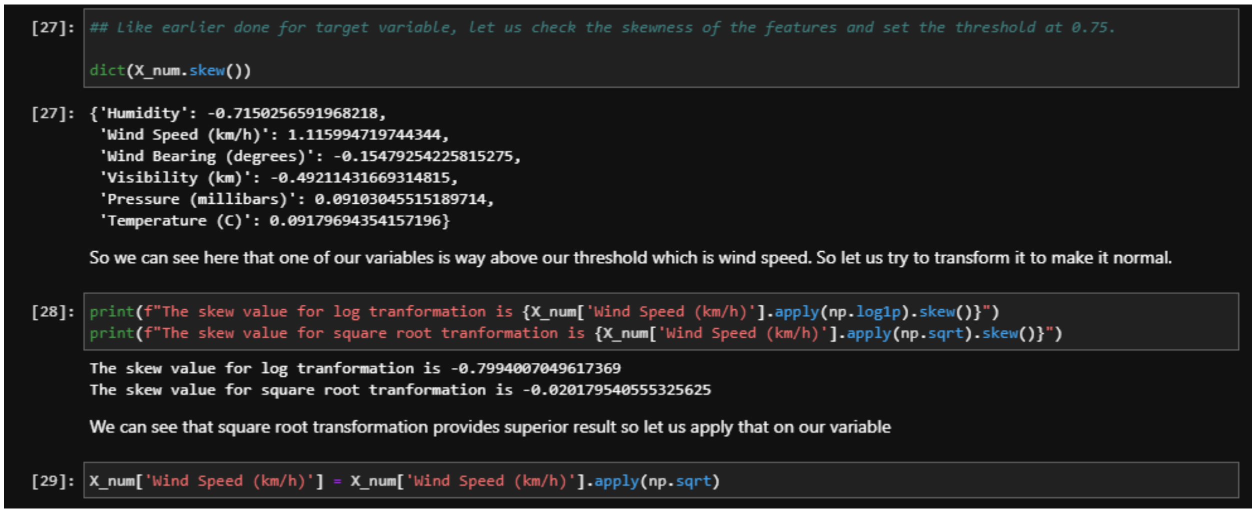
```
[19]: from scipy.stats import f_oneway
```

```
[20]: weather_summary_list = df_weather['Summary'].unique().tolist()

summary_group = [df_weather.loc[df_weather['Summary'] == summary, 'Temperature (C)'] for summary in weather_summary_list]

F, p_value = f_oneway(*summary_group)

alpha = 0.025

if p_value <= alpha:
    print(f'The null hypothesis can be rejected for values - test statistics: {test_statistics}, p_value: {p_value}')

else:
    print(f'The null hypothesis can be accepted for values - test statistics: {test_statistics}, p_value: {p_value}')
```

```
The null hypothesis can be rejected for values - test statistics: 415.970493705491, p_value: 0.0
```

Thus all categories under 'Summary' attribute can be converted into features using One-Hot Encoding

- Both categorical values were then one-hot encoded for feature generation.
- Further I checked skewness for all numerical input variables, which suggested, transformation was required for 'Wind Speed (km/h)' variable. I used Square-root transformation, as it provided best result.



```
[27]: ## Like earlier done for target variable, let us check the skewness of the features and set the threshold at 0.75.

      dict(X_num.skew())

[27]: {'Humidity': -0.7150256591968218,
       'Wind Speed (km/h)': 1.115994719744344,
       'Wind Bearing (degrees)': -0.15479254225815275,
       'Visibility (km)': -0.49211431669314815,
       'Pressure (millibars)': 0.09103045515189714,
       'Temperature (C)': 0.09179694354157196}
```

So we can see here that one of our variables is way above our threshold which is wind speed. So let us try to transform it to make it normal.

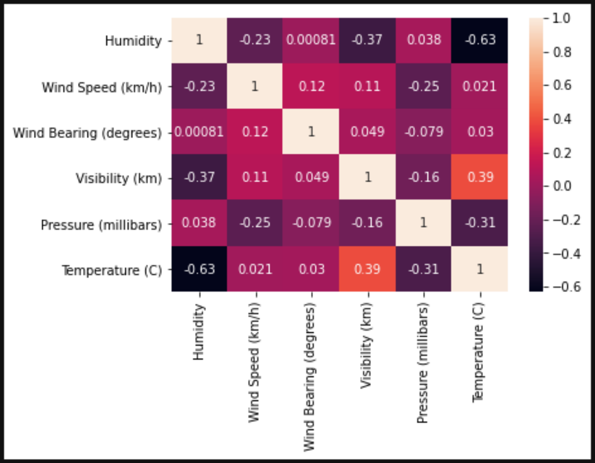
```
[28]: print(f"The skew value for log tranformation is {X_num['Wind Speed (km/h)'].apply(np.log1p).skew()}")
      print(f"The skew value for square root tranformation is {X_num['Wind Speed (km/h)'].apply(np.sqrt).skew()}")

      The skew value for log tranformation is -0.7994007049617369
      The skew value for square root tranformation is -0.020179540555325625
```

We can see that square root transformation provides superior result so let us apply that on our variable

```
[29]: X_num['Wind Speed (km/h)'] = X_num['Wind Speed (km/h)'].apply(np.sqrt)
```

- I used then Seaborn's heatmap plot for checking the correlation between each input variable. None of them were strongly correlated, thus all of them were retained for further modelling.

```
[31]: sns.heatmap(X_num.corr(), annot = True);
```



So, all our features are not strongly correlated with each other. As such we can keep them all. So finally lets add these variables to our final dataframe which we will use for modelling

4.3. Summary

- During EDA we first tried to ascertain that our target variable is normally distributed and found that it is indeed normally distributed. We used .skew() method under pandas.
- Next we went into hypothesis generation and testing process to identify whether our categorical variables explain some variance in the target variable or not.

- Eventually we kept both categorical variables and one hot encoded them to make them ready for further modelling process.
- Bigger challenge was to select our numerical features which we did in two steps first we checked how many of them are skewed and require transformation. We finally ended up transforming only Wind Speed parameter.
- In the end we checked correlations between numerical variables and found them to be within limits.
- At the end of EDA
  - ✓ number of features are 34
  - ✓ number of observations are 95936

5. Model Selection and Evaluation

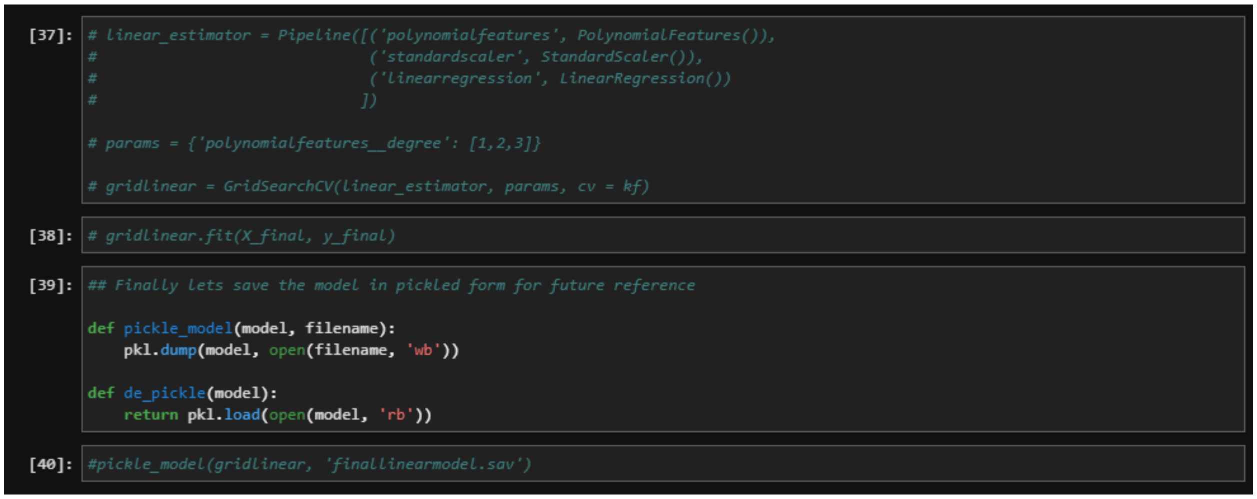
5.1. Plan

- In this part of assignment, I tried modelling using 3 types of Regression models
  - ✓ Linear Regression
  - ✓ Ridge Regression with hyperparameter (alpha) set between 0.001 to 1
  - ✓ Lasso Regression with hyperparameter (alpha) set between 0.001 to 1
- Select the best polynomial degree and hyperparameter which provides best fit for our data.
- Finally arriving on final model by reaching compromise between prediction accuracy and complexity.

5.2. Action

- For selecting best parameters, I utilized the GridSearchCV method of Sklearn module. GridSearchCV basically iterates over each input parameter range and selects the best model.
- I created pipeline containing polynomial features, standard scaler, and regression function for each model. During the process I also regularly pickled and de-pickled my models to preserve my modelling parameters during re-runs. The model iterated over following parameters
  - ✓ Polynomial Degree – 1 to 3
  - ✓ Hyper parameter (alpha) – 0.001 to 1 (only for ridge and lasso)
  - ✓ Cross validation splits – 3

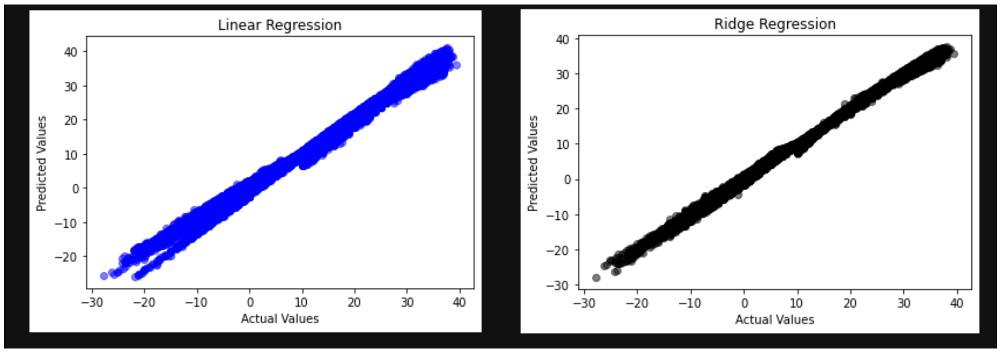
One such pipeline created for linear regression is shown below,

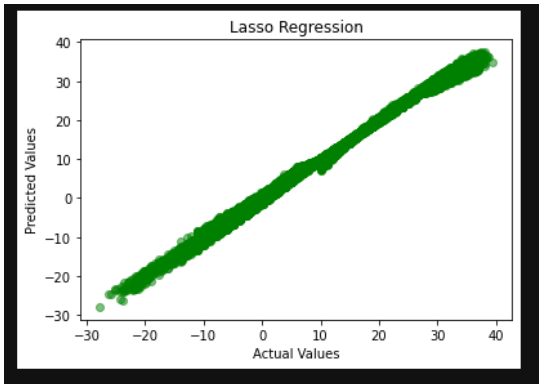


```
[37]: # linear_estimator = Pipeline([('polynomialfeatures', PolynomialFeatures()),
      #                               ('standardscaler', StandardScaler()),
      #                               ('linearregression', LinearRegression())
      #                              ])

      # params = {'polynomialfeatures__degree': [1,2,3]}

      # gridlinear = GridSearchCV(linear_estimator, params, cv = kf)

[38]: # gridlinear.fit(X_final, y_final)

[39]: ## Finally lets save the model in pickled form for future reference

      def pickle_model(model, filename):
          pkl.dump(model, open(filename, 'wb'))

      def de_pickle(model):
          return pkl.load(open(model, 'rb'))

[40]: #pickle_model(gridlinear, 'finallinearmodel.sav')
```

- For each model, the best parameters finally achieved are listed below.

	Polynomial Degree	**Alpha**	**No of Features**	**R2 Score**
Linear Regression	1	NaN	35	0.990437
Ridge Regression	3	1.000	1199	0.998019
Lasso Regression	3	0.001	213	0.997857





- From above table we can conclude that, Lasso Regression model provides a comfortable balance between the complexity and performance by providing improvement over simple linear model at the same time toning down the complexity below ridge model.

5.3. Summary

- We evaluated three model types with same initial features
  - ✓ Polynomial degree = 3
  - ✓ Cross Validation Splits = 3
- After completing the modelling we were able to achieve quiet significant results from all three models. The Optimized Model Parameters were selected through GridSearchCV method over values of alpha varying from 0.001 to 1.
- In terms of interpretation, we can very easily conclude that the factors which seem to impact apparent temperature include 'Temperature (C)', 'Wind Bearing (degrees)', 'Summary_Foggy', 'precip_snow', 'Pressure (millibars)', 'Wind Speed (km/h)', 'Humidity' and there various interaction terms to the degree 3. The order of importance of each 1st degree feature in absolute terms is shown below.

Feature	**Coefficient**
Temperature (C)	8.474063
Wind Bearing (degrees)	0.057787
Summary_Foggy	-0.054222
precip_snow	-0.051797
Pressure (millibars)	0.016961
Wind Speed (km/h)	-0.016721
Humidity	0.003518

- However, we should be careful while interpreting these results - since apparent temperature is calculated using a well-established scientific formula. Not to forget we finalized a lasso model which deletes certain features which raises uncertainty on our interpretation.
- Nevertheless, our model has indeed demonstrated a very good prediction power, which was our primary objective, and thus, can be used to estimate future values quite reliably.

6. Conclusion and Future Directions

6.1. In conclusion, we chose Lasso Regression Model for purpose of deployment for future predictions. The model not only shows great prediction power, it is also relatively simple.

6.2. In future,

- We can attempt to re-train the lasso model for even higher values of iteration (I had to limit my iteration to 5000, due to insufficient memory) and alpha parameters of ridge and lasso regression.
- We may also try to eliminate categorical variables entirely and check their impact on the model.
- Attempting modelling using any higher degree of polynomial may improve the accuracy but at the cost of complexity (thereby overfitting the data).
- Another thing that is required to be kept in mind is that the model depends on certain categorical values (features), thereby it is important to ensure that those categorizations are done based on well-defined parameters. Any time those parameters change we will be required to accordingly re-train our model based on those updated categories.