

Weather Prediction

1. Project Title : Weather Prediction

2. Abstract :

The activities of many primary sectors depend on the weather for production, e.g. farming. The climate is changing at a drastic rate nowadays, which makes the old weather prediction methods less effective and more hectic. To overcome these difficulties, the improved and reliable weather prediction methods are required. These predictions affect a nation's economy and the lives of people. To develop a weather forecasting system that can be used in remote areas is the main motivation of this work. The data analytics and machine learning algorithms, such as random forest classification, are used to predict weather conditions. Weather forecasting can be done by "Using Linear Regression" or "Neural Network models" and many more. The main aim of this project is to implement a weather forecasting technique which is a low-cost and portable solution using "Linear Regression" model.

3. Dataset :

The data used in this Project is collected from Weather Underground's free tier API web service. The company provides a swath of API's that are available for both commercial and non-commercial uses. The history API provides a summary of various weather measurements for a city and state on a specific day. To interact with the API to pull in weather data of Jaipur, request libraries are used. Once collected, the data will need to be processed and aggregated into a format that is suitable for data analysis, and then cleaned.

4. Prepare Data :

- **Pre-Processing :**

First a request is made to the Weather Underground history API and then processing of the returned data is made using a few standard libraries as well as some popular third party libraries. The request for the dataset is made in two batches with each batch making 500 requests.

Now that the requested dataset is a sizable records list of DailySummary named tuples we will use it to build out a Pandas DataFrame. "Pandas.DataFrame(...)" class constructor is used to instantiate a DataFrame object. The parameters passed to the constructor are records which represent the data for the DataFrame, the features list is also used to define the "DailySummary" named tuples which will specify the columns of the DataFrame. The set_index() method is chained to the DataFrame instantiation to specify date as the index.

It is quite helpful to have subject matter knowledge in the area under investigation to aid in selecting meaningful features to investigate paired with a thoughtful assumption of likely patterns in data. For this project we have selected quite a few features while parsing the returned daily summary data to be used in our study are :

- mean temperature
- mean dewpoint
- mean pressure
- max humidity
- min humidity
- max dewpoint
- min dewpoint
- max pressure
- min pressure
- Precipitation

Now these new features as columns in DataFrame. To do so a smaller subset of the current DataFrame is made. And a "tmp" DataFrame consisting of just 10 records and the features "meantempm" and "meandewptm" is made.

After the DataFrame is made for each day (row) and for a given feature (column) we would like to find the value for that feature N days prior. For each value of N (1-3 in our case) we want to make a new column for that feature

representing the Nth prior day's measurement.

As the basic steps required to make the new features are available. Now these steps are wrapped up into a reusable function and put it to work building out all the desired features.

Data Cleaning :

To ensure the quality of the data for this project, identifying unnecessary data, missing values, consistency of data types, and outliers then making some decisions about how to handle them if they arise are very important.

The first thing done is dropping any columns of the DataFrame that are not required to reduce the amount of data which we are working with. The goal of the project is to predict the future temperature based on the past three days of weather measurements; it only requires min, max, and mean temperatures for each day plus all the new derived features mentioned above.

Next is by making use of some built in Pandas functions to get a better understanding of the data. The first function used is a DataFrame method called "info()" which provides information on the DataFrame. Of interest is the "data type" column of the output.

As the data type of every column is of type "object". All the feature columns are converted into "float" for the type of numerical analysis that we hope to perform. "apply()" DataFrame method is used to apply the Pandas "to_numeric" method to all values of the DataFrame.

Now that all of our data has the data type required observation of some summary stats of the features and use the statistical rule of thumb to check for the existence of extreme outliers are done. The DataFrame method "describe()" is used to produce a DataFrame containing the count, mean, standard deviation, min, 25th percentile, 50th percentile (or median), the 75th percentile and the max value. As this is very useful information to evaluate the distribution of the feature data.

Another Information is added by calculating another output column,

indicating the existence of outliers. The rule of thumb to identifying an extreme outlier is a value that is less than 3 interquartile ranges below the 25th percentile, or 3 interquartile ranges above the 75th percentile. Interquartile range is simply the difference between the 75th percentile and the 25th percentile.

The first set of features all appear to be related to max humidity. As we can observe that the outlier for this feature category is due to the apparently very low min value using graphical way by taking histogram. The histogram of the values for “maxhumidity” the data exhibits quite a bit of negative skew. Many of the underlying statistical methods assume that the data is normally distributed.

Next feature is minimum pressure feature distribution and is plotted in histogram. The plot exhibits another interesting feature. From this plot, the data is multimodal, which leads that there are two very different sets of environmental circumstances apparent in this data. The final category of features containing outliers, precipitation. Since the dry days (ie, no precipitation) are much more frequent, it is sensible to see outliers.

The last data quality issue to address is that of missing values. Due to the way in which we have built out the DataFrame, the missing values are represented by NaNs. There is a column of output that lists the non-null values for each feature column. Looking at this information you can see that for the most part the features contain relatively few missing (null / NaN) values, mostly just the ones that are introduced. However, the precipitation columns appear to be missing a significant part of the data.

Missing data poses a problem because most machine learning methods require complete data sets devoid of any missing data. Aside from the issue that many of the machine learning methods require complete data, if I were to remove all the rows just because the precipitation feature contains missing data then I would be throwing out many other useful feature measurements. The missing values are filled with an interpolated value that is a reasonable estimation of the true values.

Since preserving as much of the data as possible, where there is minimal risk of introducing erroneous values, the missing values are filled with precipitation values with the most common value of zero. As this is a reasonable decision because the great majority of values in the precipitation measurements are zero.

Now that all the missing values are filled , while being cautious not to negatively impact the quality, we would be comfortable simply removing the remaining records containing missing values from the data set. As it is quite easy to drop rows from the DataFrame containing NaNs. All that is done is call the method `dropna()` and Pandas will do all the work for me.

At last as all the required features are introduced and the data is clean we converted it into a “CSV” file named “Weather.”

- **Summarization:**

To understand about the data DataFrame method called `info()` is used which provides information on the DataFrame. The DataFrame method `describe()` is also used which will produce a DataFrame containing the count, mean, standard deviation, min, 25th percentile, 50th percentile (or median), the 75th percentile and, the max value. This can be very useful information to evaluate the distribution of the feature data.

The size of the Dataset used is 26364 and the dimensions are (676, 39).

- **Data Visualization :**

The plotting of the data is done between the dependent variable "meantemp" be the consistent y-axis along all of the 18 predictor variables plots. We accomplished this by creating a grid of plots. Pandas is used which comes with a useful plotting function called the `scatter_plot()`. A grid structure is created with six rows of three columns in order to avoid sacrificing clarity in the graphs. And correlation of the data is also plotted.

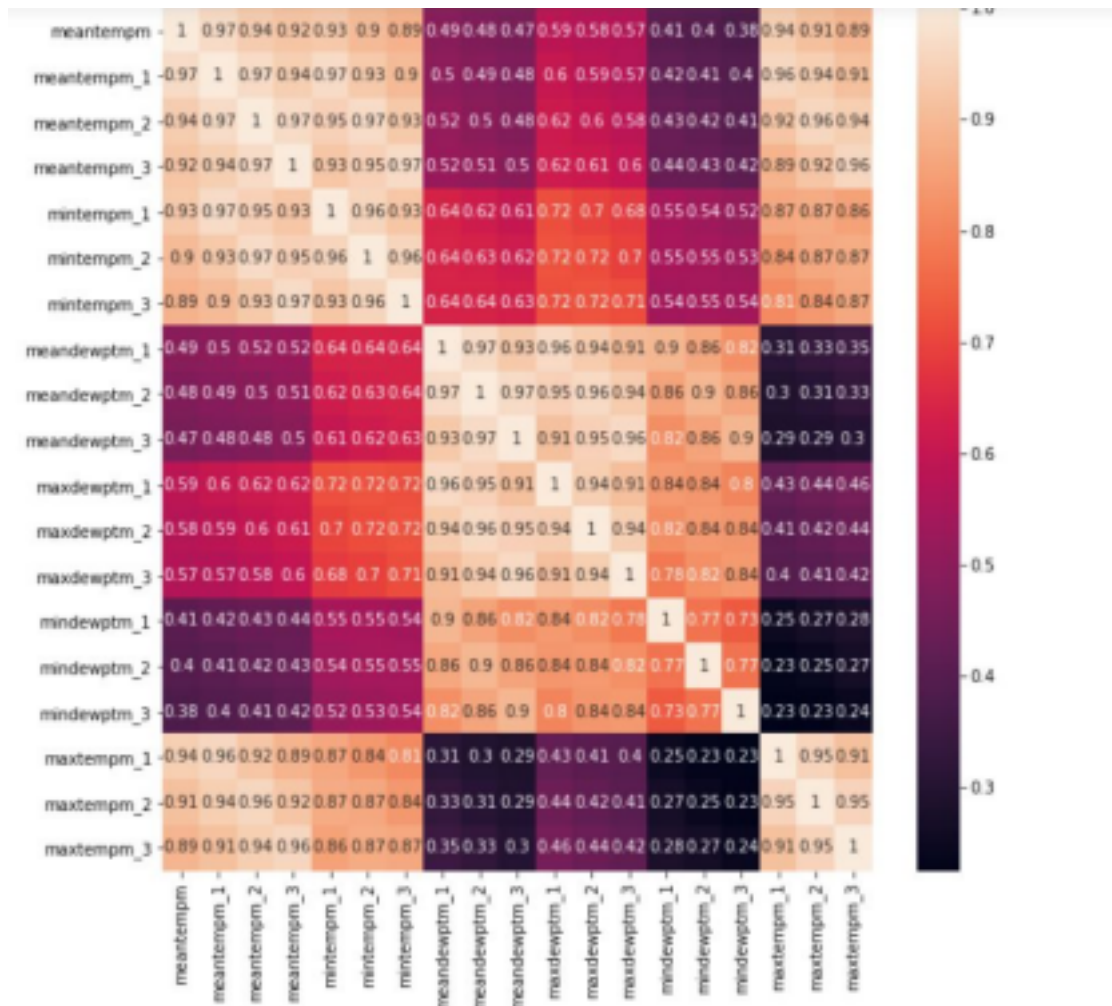


Fig-1 : Correlation Plotting

From the plots it can be described that all the remaining predictor variables show a good linear relationship with the response variable ("meantempm"). Additionally, it is also worth noting that the relationships all look uniformly randomly distributed. By this it means there appears to be relatively equal variation in the spread of values devoid of any fanning or cone shape.

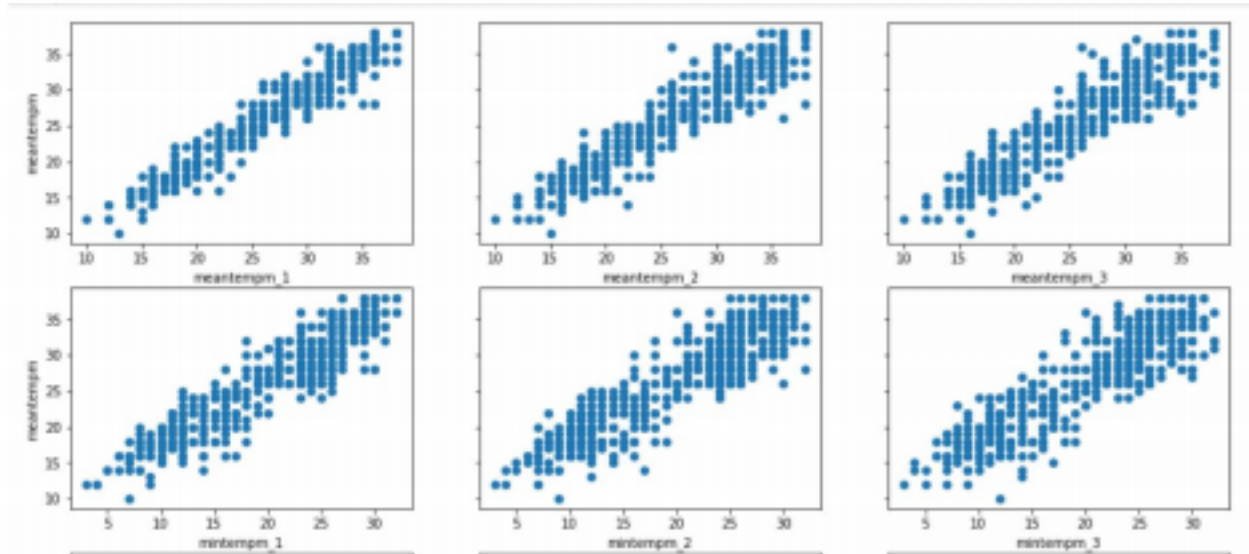


Fig - 2 : Visualisation of Data

5. Python packages :

LIBRARIES	DESCRIPTION
datetime	The datetime module supplies classes for manipulating dates and times. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.
time	Used to delay requests to stay under 10 per minute
collections	This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, dict, list, set, and tuple.
pandas	It is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and

	operations for manipulating numerical tables and time series.
requests	Make a request to a web page, and print the response text
matplotlib	Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.
numpy	NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices.
statsmodels	It is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct.
sklearn	Scikit-learn is one of the most useful libraries for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
seaborn	Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

6. Algorithm used :

The Algorithm used is Linear Regression model to predict future mean daily temperature values based on the dataset used. To build the Linear Regression model we use two important Python libraries in the Machine Learning industry: Scikit-Learn and StatsModels.

● **Background on Linear Regression using Ordinary Least Squares**

:

Linear regression aims to apply a set of assumptions primary regarding linear relationships and numerical techniques to predict an outcome (Y, aka the dependent variable) based off of one or more predictors (X's independent variables) with the end goal of establishing a model (mathematical formula) to predict outcomes given only the predictor values with some amount of uncertainty.

The generalized formula for a Linear Regression model is:

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_{(p-n)} x_{(p-n)} + E$$

where:

- \hat{y} - is the predicted outcome variable (dependent variable)
- x_j - are the predictor variables (independent variables) for $j = 1, 2, \dots, p-1$
parameters
- β_0 - is the intercept or the value of \hat{y} when each x_j equals zero •
- β_j - is the change in \hat{y} based on a one unit change in one of the corresponding x_j
- E - is a random error term associated with the difference between the predicted \hat{y}_i value and the actual y_i value.

That last term in the equation for the Linear Regression is a very important

one. The most basic form of building a Linear Regression model relies on an algorithm known as Ordinary Least Squares which finds the combination of β_j 's values which minimize the E term.

- **Selecting Features for our Model :**

A key assumption required by the linear regression technique is that you have a linear relationship between the dependent variable and each independent variable. One way to assess the linearity between our independent variable, which for now will be the mean temperature, and the other independent variables is to calculate the Pearson correlation coefficient.

The Pearson correlation coefficient (r) is a measurement of the amount of linear correlation between equal length arrays which outputs a value ranging -1 to 1. Correlation values ranging from 0 to 1 represent increasingly strong positive correlation. By this I mean that two data series are positively correlated when values in one data series increase simultaneously with the values in the other series and, as they both go up in increasingly equal magnitude the Pearson correlation value will approach 1.

Correlation values from 0 to -1 are said to be inversely, or negatively, correlated in that when the values of one series increase the corresponding values in the opposite series decrease but, as changes in magnitude between the series become equal (with opposite direction) the correlation value will approach -1. Pearson correlation values that closely straddle either side of zero are suggestive to have a weak linear relationship, becoming weaker as the value approaches zero.

To assess the correlation in this data `corr()` method of the Pandas DataFrame object is called. Chained to this `corr()` method call I can then select the column of interest ("meantemp") and again chain another method call `sort_values()`

on the resulting Pandas Series object. This will output the correlation values from most negatively correlated to the most positively correlated.

In selecting features to include in this linear regression model, we would like to error on the side of being slightly less permissive in including variables with moderate or lower correlation coefficients. So we will be neglecting the features that have correlation values less than the absolute value of 0.6. Also, since the "mintemp" and "maxtemp" variables are for the same day as the prediction variable "meantemp", is also removed.

With this information, A new DataFrame is created that only contains the variables of interest.

● **Using Step-wise Regression to Build a Robust Model :**

A model is considered to be robust if its output and forecast are considered accurate even if one or more of the input variables or assumptions are changed due to unforeseen circumstances.

In this project robust Linear Regression model is utilizing statistical tests for selecting meaningful, statistically significant, predictors which are selected by utilising python "statsmodels" library.

The reason behind selecting this model is one of its hypothesis test is to evaluate the significance of each of the included predictor variables. The formal definition of the hypothesis test for the significance of a β_j parameters are as follows:

- $H_0: \beta_j = 0$, the null hypothesis states that the predictor has no effect on the outcome variable's value
- $H_a: \beta_j \neq 0$, the alternative hypothesis is that the predictor has a significant effect on the outcome variable's value

The tests of probability to evaluate the likelihood that each β_j is significant beyond simple random chance at a selected threshold α we can be more stringent in selecting the variables to include resulting in a more robust model.

Step-wise regression is to test for the effects of interactions on the significance of any one variable in a linear regression model. Using the we can add or remove variables from the model and assess the statistical significance of each variable on the resultant model.

The technique used in this is “Backward Elimination” as it is a fully loaded general model that includes all the variables of interest. Steps followed for Backward Elimination are as follows :

1. Select a significance level α for which you test your hypothesis against to determine if a variable should stay in the model
2. Fit the model with all predictor variables
3. Evaluate the p-values of the β_j coefficients and for the one with the greatest p-value, if p-value $> \alpha$ progress to step 4, if not you have your final model
4. Remove the predictor identified in step 3
5. Fit the model again but, this time without the removed variable and cycle back to step 3.

By following the above steps a model is built using the “statsmodels”.

Next a call named “summary()” is called and the information about the data is shown however for this model we will be using only about 2-3 values which are selected by following the below rules:

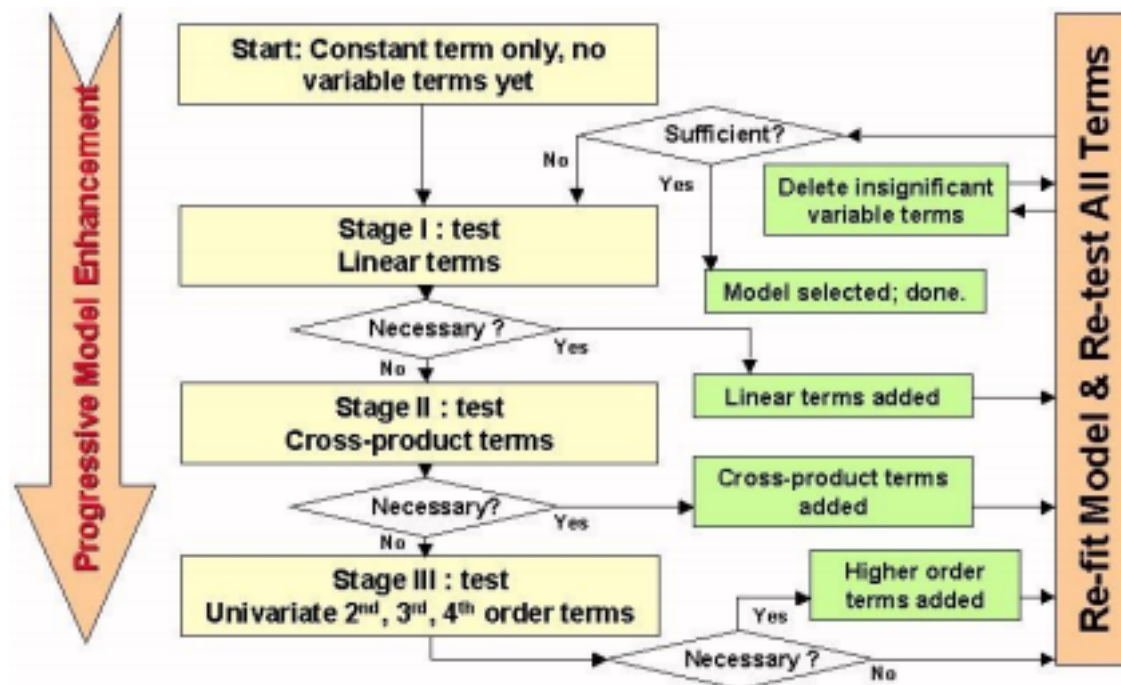
1. $P > |t|$ - this is the p-value I mentioned above that I will be using to evaluate the hypothesis test. This is the value we are going to use to determine whether to eliminate a variable in this step-wise backward elimination technique.
2. R-squared - a measure that states how much of the overall variance in the outcome our model can explain
3. Adj. R-squared - the same as R-squared but, for multiple linear regression

this value has a penalty applied to it based on the number of variables being included to explain the level of overfitting.

Two things that are to be noted will doing backwards elimination technique are:

(1) the R-squared and Adj. R-squared values are both equal which suggests there is minimal risk that our model is being over fitted by excessive variables.

(2) the value of 0.894 is interpreted such that our final model explains about 90% of the observed variation in the outcome variable, the "meantempm".



- **Using SciKit-Learn's LinearRegression Model to Predict the Weather :**

First the dataset is split into a testing set and training set by importing the "train_test_split()" function from "sklearn.model_selection". And the training and testing datasets are split into 80% training and 20% testing and assign a "random_state" of 12 for getting the random selection of the data.

Next step is building the regression model using the training dataset taken in the above step by importing the "LinearRegression" class from the "sklearn.linear_model" module. As, scikit-learn scores major usability bonus points by implementing a common fit() and predict().

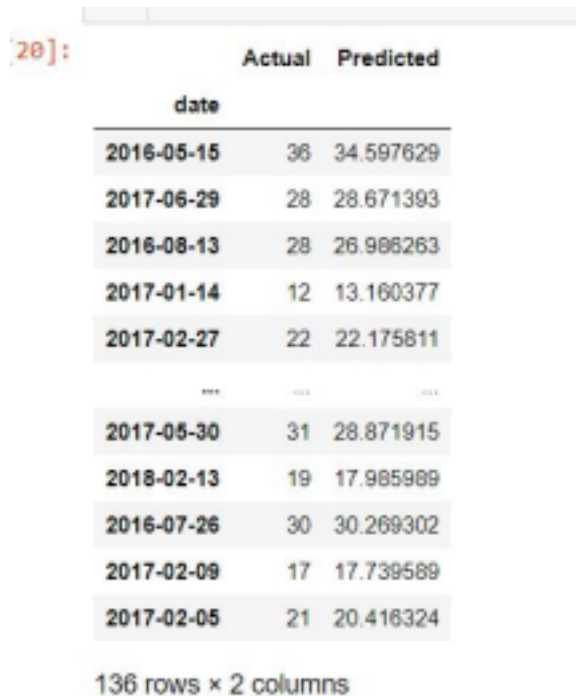
Next step for gaining an interpretative understanding of the model

validity regressor model's "score()" function is used to determine that the model is able to explain about 90% of the variance observed in the outcome variable, mean temperature.

Next the "mean_absolute_error()" and "median_absolute_error()" of the "sklearn.metrics" are used to determine that on average the predicted value.

7.Experimental Result obtained :

At the end of the project we were able to build a rigorous Linear Regression model to predict the future mean daily temperature values.



```
[20]:
```

	Actual	Predicted
date		
2016-05-15	36	34.597629
2017-06-29	28	28.671393
2016-08-13	28	26.986263
2017-01-14	12	13.160377
2017-02-27	22	22.175811
...
2017-05-30	31	28.871915
2018-02-13	19	17.985989
2016-07-26	30	30.269302
2017-02-09	17	17.739589
2017-02-05	21	20.416324

136 rows x 2 columns

Fig - 3 : Result-1

It is also determined in this module that on average the predicted value is about 3 degrees Celsius off and half of the time it is off by about 2 degrees

Celsius when the variance, Mean Absolute Error and Media Absolute Error are calculated.

```
The Explained Variance: 0.95
The Mean Absolute Error: 1.11 degrees celsius
The Median Absolute Error: 0.91 degrees celsius
```

Fig - 4 : Result - 2

8. Conclusion :

In this project it is described how to collect, clean and process a reasonably good-size data set which is used for our regression model. And through this project it is also understood how to use the Linear Regression Machine Learning algorithm to predict future mean weather temperature based on the data which is collected, cleaned and processed. And through this project we can know how to use the “statsmodels” library to select statistically significant predictors based on sound statistical methods. Then utilising the information we also fit a prediction model based on a training subset using Scikit-Learn’s “Linear Regression” class. Predicting the expected values based off of the inputs from a testing subset and evaluate the accuracy of the prediction by using a fitted model which is helpful to indicate the reasonable amount of accuracy.