CENG313
INTRODUCTION TO DATA SCIENCE

FINAL REPORT

171180003 Beyza Akkoyun

161180038 Hilal Ilgaz

ABSTRACT

Machine learning and artificial intelligence fields has given rise to countless weather prediction models. But the problem of exactly predicting or forecasting the weather still continues. Numerical weather prediction is taking the numerical data on weather conditions and applying machine learning algorithms on it to the weather forecasting. This project is the application of machine learning algorithms such as random forest classifier and decision tree classifier, linear regression model, logistic regression model from statistics. The methods which are used in the project are modified in order to obtain the most optimum error percentage by iterating and adding some percentage of error to the input values. The methods are applied on the set of data and the coefficients are used to predict the weather forecasting. based on the corresponding values of the parameters.

Keywords- Weather prediction, machine learning, linear regression, random forest classifier, decision tree classifier, logistic regression.

1. INTRODUCTION

The application of science and technology that predicts the state of atmosphere at any given particular time period is known as Weather forecasting. There is a many different methods to weather forecast. Weather forecast notices are important because they can be used to prevent demolition of life and environment. Weather conditions are required to be predicted not only for future planning in farming and industries but also in many other fields like defence, mountaineering, shipping and aerospace navigation etc. It is often used to warn about natural disasters are caused by sudden change in climatic conditions [1].

Machine learning is the ability of computer to learn without being explicitly programmed. It allows machines to find hidden patterns and insights. In supervised learning, we build a model based on labeled training data. The model is then used for mapping new examples. So, based on the observed weather patterns from the past, a model can be built and used to predict the weather. The training of the data is done using a modified version of Linear Regression, , Random Forest Classifier, Decision Tree Classifier and Logistic Regression methods. The error percentage between the actual and predicted is used to improve the training set and train the data with the new inputs [1].

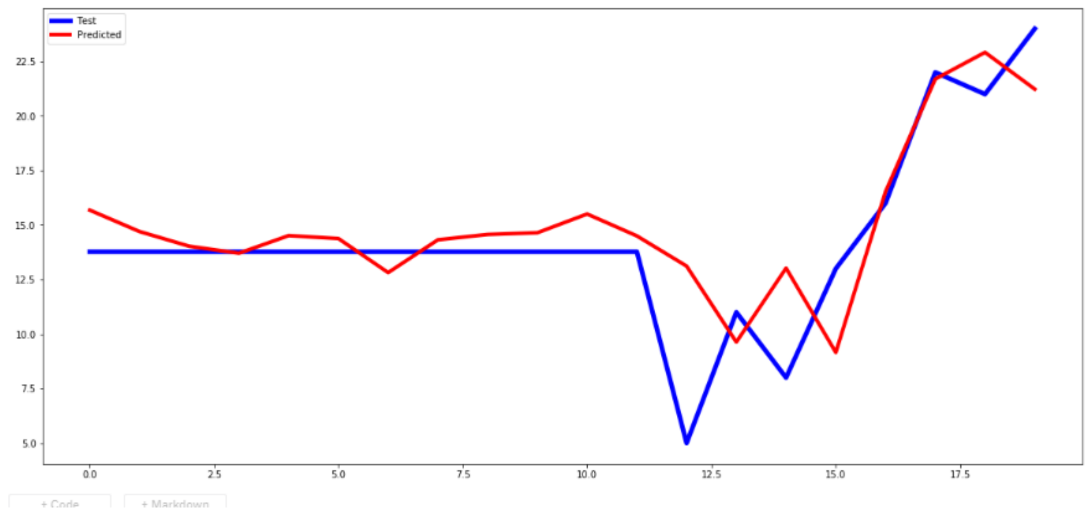
In this project Linear Regression, Random Forest Classifier, Decision Tree Classifier, Logistic Regression are used to develop a model for forecasting weather parameters. The proposed model is capable of forecasting the weather conditions for a particular station using the collected data. With this project, weather can be forecasted with greater accuracy, which will be helpful in daily activities.

2. RELATED WORK

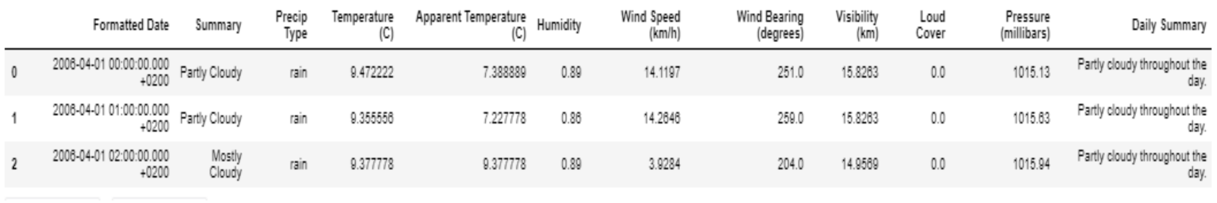
Weather is one of the most influential factors that decide the life activity of the members of the ecosystem, which influence the economic factor of country and people of the region. To eliminate the disaster due to weather, an activity has to be activated to predict the weather uncertainty behavior in future. Researchers have started highlighting the effectiveness of data mining algorithms in predicting the weather. One of the latest research works includes a paper [9] by Ashwini Mandale, Jadhawar B.A. this paper makes a mention of Artificial Neural Networks and Decision Tree algorithms and their performance in prediction of weather. ANN finds a relationship between the weather attributes and builds a complex model, whereas decision tree learns the trend of data and accordingly builds a classifier tree that can be used for prediction. The other research [5] by Wai Naing, Zaw Htike makes forecasting with using random forest algorithm. Accuracy score is %55,97. Their model has a good performance and reasonable prediction accuracy. Their model's forecasting reliabilities were evaluated by computing the mean absolute error and root mean square error between the exact and predicted values. Paras and Sanjay [10] developed a forecasting model using mathematical regression. The weather data is collected for a period of 3 years and this model can predict max and min temperatures for a period of 15 to 45 weeks into the future. They used Multiple Linear Regression. It is similar Logistic Regression. Their accuracy score is 50.0%.

3. DATASET

First of all our dataset was İstanbul Weather Forecast, it collected from a website called "World Weather Online". The case data covered to period of 2009 to 2019. We started with this dataset but when learning started we realized our dataset lots of repetitive row. It was showed prediction its graphic looks like



After this graphic we decided to change our dataset. Our new dataset is Szeged, Hungary Weather Data. The dataset covered the period of 2006-2016. It is collected from DarkSky website. It has 96453 rows and 14 columns. It is a huge dataset but nearly clean dataset. Top three rows and all columns



	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.

3.1. Data Analysis and Cleaning

We analyzed out dataset by dividing it into two section. These are categorical variables and quantitative variables.

```
categorical = df.select_dtypes(include = ["object"]).keys()
```

```
Index(['Formatted Date', 'Summary', 'Precip Type', 'Daily Summary'], dtype='object')
```

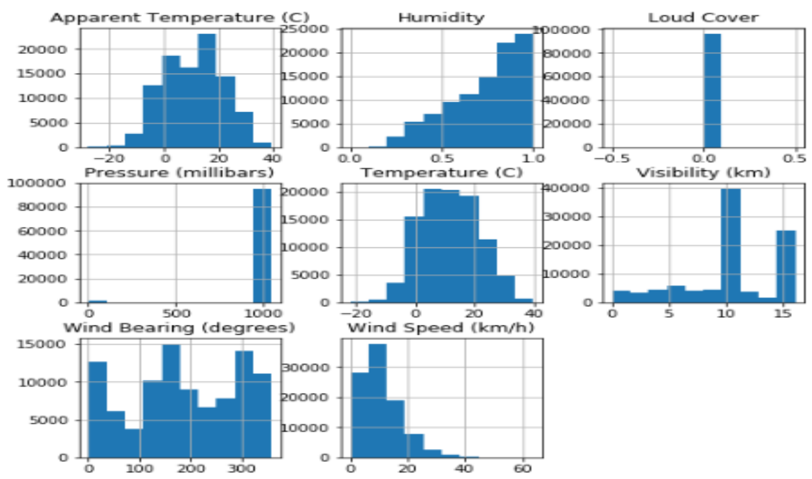
```
quantitative = df.select_dtypes(include = ["int64","float64"]).keys()
```

```
Index(['Temperature (C)', 'Apparent Temperature (C)', 'Humidity',
       'Wind Speed (km/h)', 'Wind Bearing (degrees)', 'Visibility (km)',
       'Loud Cover', 'Pressure (millibars)'],
      dtype='object')
```

After this dividing process, we splitted to Formatted Date field into groups. Our dataset's info is added three more columns.

```
#'Formatted Date' transformation:
df['Date'] = pd.to_datetime(df['Formatted Date'], utc=True, format='%Y%m%d %H')
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day
df['hour'] = df['Date'].dt.hour
```

When our date transformation finished, we started to making our dataset visualize. Our quantitative variable looked like this:



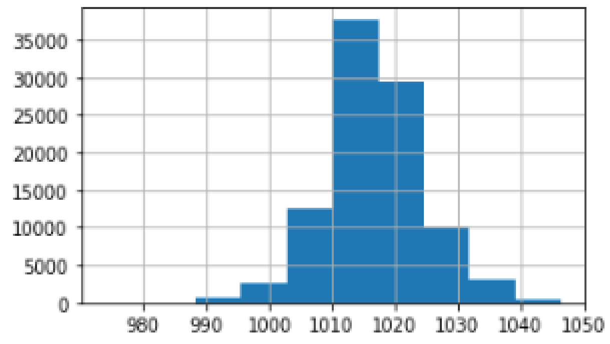
According to column distributions, ‘Loud Cover’ takes values zero so we drop it.

```
df=df.drop('Loud Cover',axis=1)
```

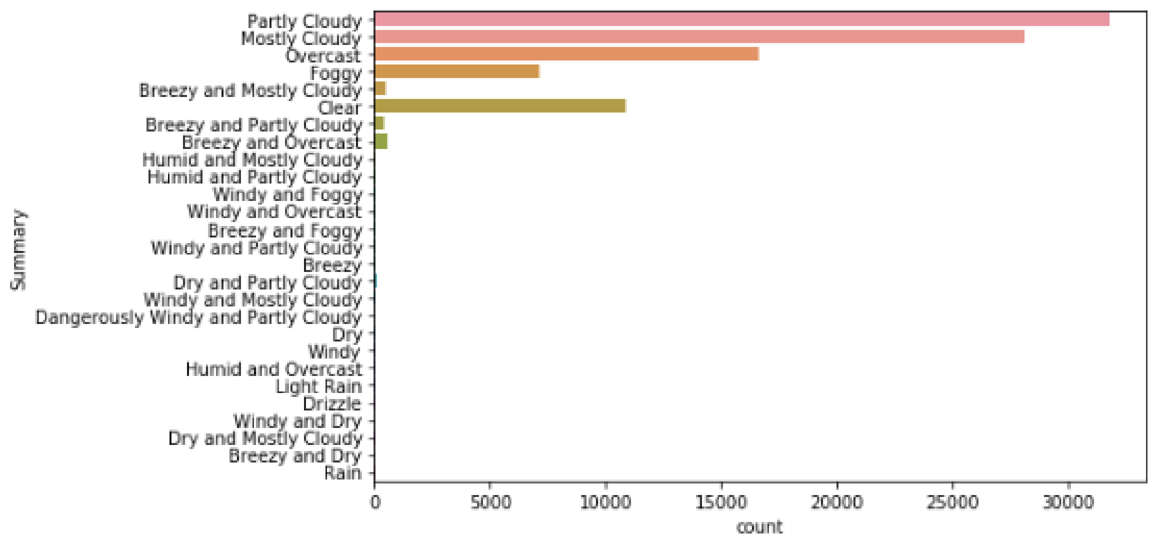
In ‘Pressure (millibars)’ field, some observations are zero. It means that they are missing values. We filled the zeros with the median for rational results.

```
pressure_median = df['Pressure (millibars)'].median()
def pressure(x):
    if x==0:
        return x + pressure_median
    else:
        return x
df["Pressure (millibars)"] = df.apply(lambda row:pressure(row["Pressure (millibars)"]) , axis = 1)
rcParams['figure.figsize'] = 5, 3
df['Pressure (millibars)'].hist()
```

After this function Pressure (millibars) column distrubution:



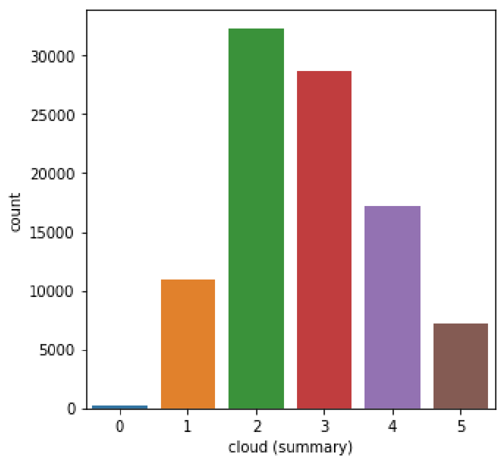
Analyzing the categorical variables started with ‘Summary’ column. There are 27 ‘Summary’ categories which makes it a bit confusing. We looked which are the most common.



```
summary_freq=pd.crosstab(index=df['Summary'],columns="count")
summary_freq_rel = summary_freq/summary_freq.sum()
summary_freq_rel.sort_values('count', ascending=False) #relative frequencies
```

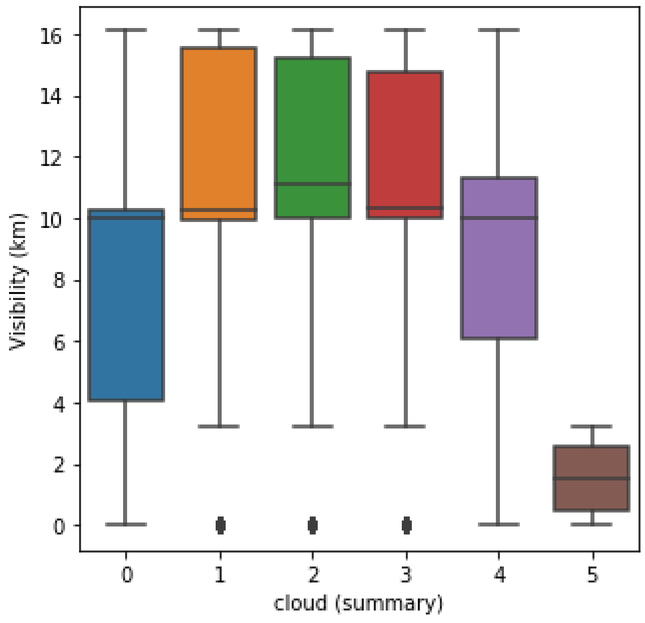
According to this code, most common categories are: Partly Cloudy (33%), Mostly Cloudy(29%), Overcast (17%), Clear (11%), Foggy(7%) = 97% out of the total. Moreover, these weather conditions also appear with other weather characteristics. E.g. 'Windy and Foggy'. We created a new categorical variable 'Cloud (summary): Foggy (5), Overcast (4), Mostly Cloudy (3), Partly Cloudy (2), Clear (1), Nothing (0)' because we wanted to Show that the all columns in dataset is related. There are other weather characteristics contained in 'Summary': Windy&Breezy, Rain, Humid&Dry but there are just a few observations within these categories. Moreover this information is already contained in other variables: 'Wind Speed (km/h)', 'Precip Type' and 'Humidity'.

```
def cloud_categorizer(row):
    row = str(row).lower()
    category = ""
    if "foggy" in row:
        category = 5
    elif "overcast" in row:
        category = 4
    elif "mostly cloudy" in row:
        category = 3
    elif "partly cloudy" in row:
        category = 2
    elif "clear" in row:
        category = 1
    else:
        category = 0
    return category
df["cloud (summary)"] = df.apply (lambda row:cloud_categorizer(row["Summary"]) , axis = 1)
```



As can be observed, there are a few observations with no information about "clouds" or foggy in variable 'Summary'. We decided to look at variable 'Visibility (km)', which obviously is correlated to "clouds" and foggy.

```
sns.boxplot(x=df['cloud (summary)'], y=df['Visibility (km)'])
```



It seems that the boxplot of the null values is very similar to the one of "overcast"so we assumed that obsevations with no information about "clouds" and foggy fall in the "overcast" category. We analyzed ‘Summary’ column with all of details. After that, we continued with ‘Daily Summary’ column. There are 214 unique categories in that column, the most ten common categories found with the code.

```
daily_summary_freq =pd.crosstab(index=df['Daily Summary'],columns="count")
daily_summary_freqrel=daily_summary_freq/daily_summary_freq.sum()
daily_summary_freqrel.sort_values('count', ascending=False).head(10)
```

col_0	count
Daily Summary	
Mostly cloudy throughout the day.	0.208236
Partly cloudy throughout the day.	0.103480
Partly cloudy until night.	0.063959
Partly cloudy starting in the morning.	0.053746
Foggy in the morning.	0.043555
Foggy starting overnight continuing until morning.	0.037075
Partly cloudy until evening.	0.034089
Mostly cloudy until night.	0.032088
Overcast throughout the day.	0.030606
Partly cloudy starting in the morning continuing until evening.	0.029092

We started to counting of values for finding null values.

```
data['Daily Summary'].value_counts(dropna=False)
data['Summary'].value_counts(dropna=False)
data['Precip Type'].value_counts(dropna=False)
```

After counnting just 'Precip Type' column has null values.

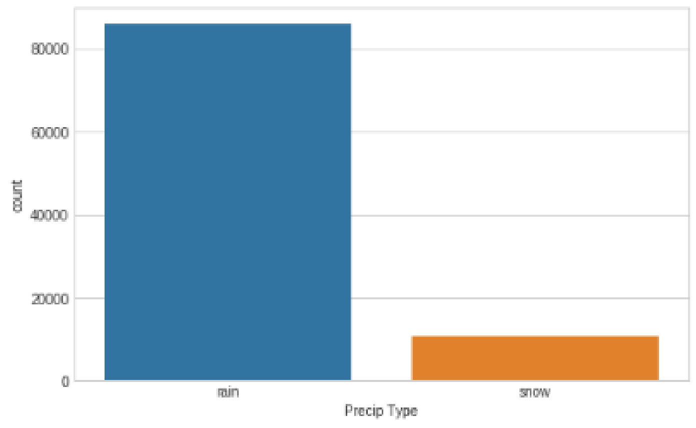
```
rain     85224
snow     10712
NaN        517
Name: Precip Type, dtype: int64
```

We filled the null values. Our 'Precip Type' column doesn't have null values anymore.

```
data.fillna(method='ffill', inplace=True)
sns.countplot(x=data['Precip Type'])
```

Except this package function we tried another solution.

```
for index,row in df_filtered[df_filtered['Precip Type'].isnull()].iterrows():
    most_frequent = df_filtered[df_filtered["year"] == row["year"]]['Precip Type'].value_counts().idxmax()
    df_filtered.at[index, "Precip Type"] = most_frequent
# Lets replace missing values in _dewptm. We can take an avergae of that year
df_filtered[df_filtered['Precip Type'].isnull()]
```



Because of this process we looked missing values.

```
# Calculate total number of cells in dataframe
totalCells = np.product(data.shape)
# Count number of missing values per column
missingCount = data.isnull().sum()
# Calculate total number of missing values
totalMissing = missingCount.sum()
# Calculate percentage of missing values
print("The weather history dataset contains", round(((totalMissing/totalCells) * 100), 2), "%", "missing values.")
```

```
The weather history dataset contains 0.0 % missing values.
```

We don't have missing values. For analyzing we visualized the average temperature. We calculated every years' average temperature.

```
# Average temprature
print("average temprature in szeged:", round(df['Temperature (C)'].mean(axis=0),2))
```

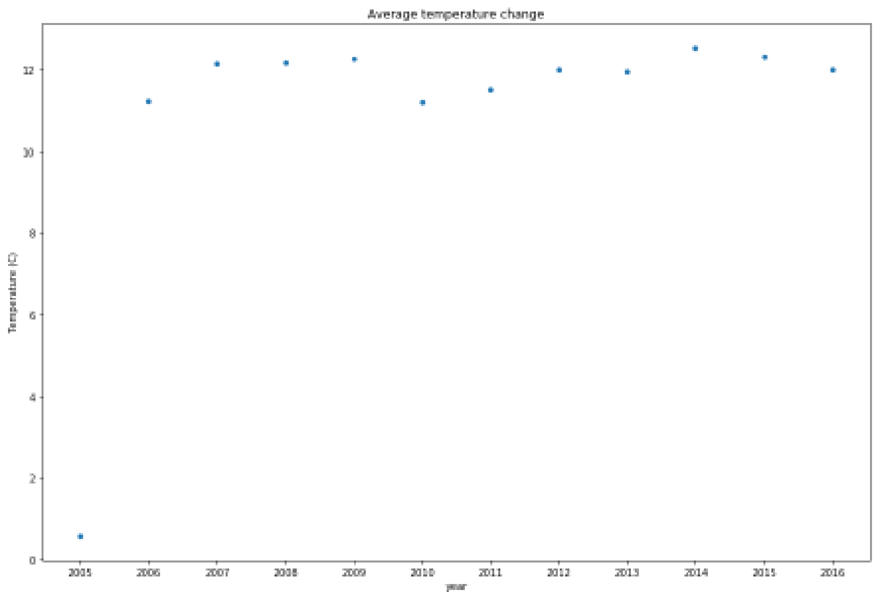
```
average temprature in szeged: 11.93
```

In our dataset we have December in 2005. This is the only data incluing that year. So average temperature calculations continue with that data.

```
df['Temperature (C)'].groupby(df.year).mean()
```

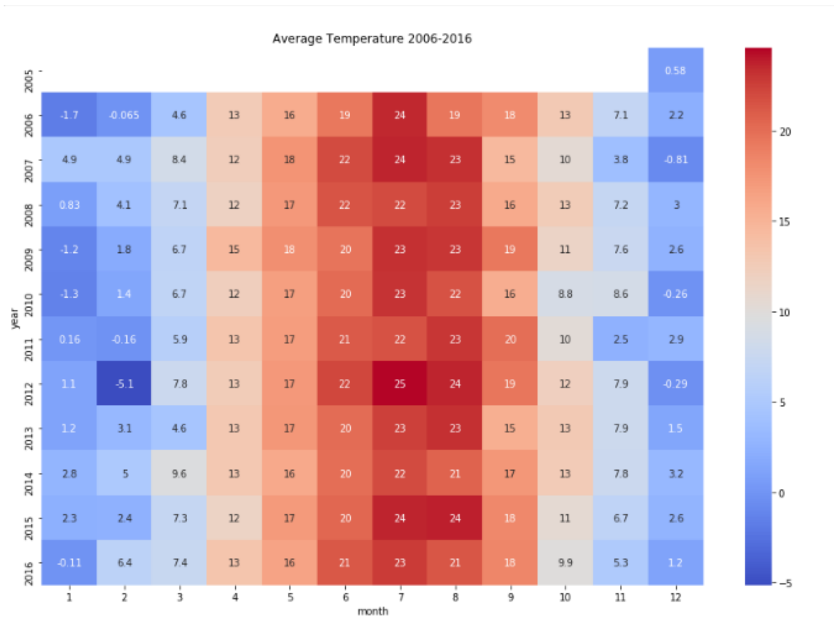
```
year
2005     0.577778
2006    11.215225
2007    12.134677
2008    12.161819
2009    12.269682
2010    11.200176
2011    11.524934
2012    11.986824
2013    11.941017
2014    12.528228
2015    12.312088
2016    11.987381
Name: Temperature (C), dtype: float64
```

Average temperature changes looks like:



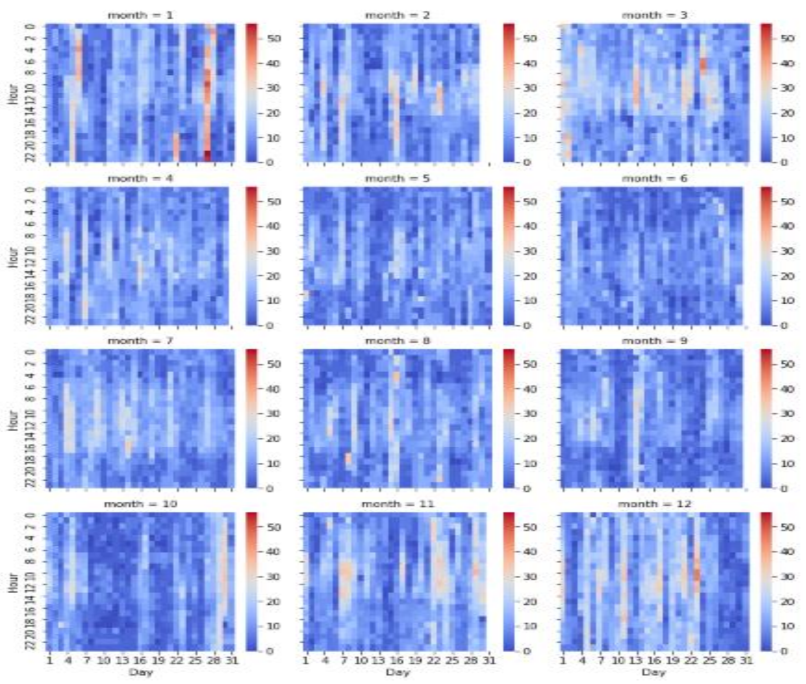
We use heat map for understandable average temperature change.

```
# Heatmap for year and average temperature across the month. More red more heat, more blue less heat
plt.figure(figsize=(15, 10));
sns.heatmap(pd.crosstab(df_filtered.year, [df_filtered.month], values=df_filtered['Temperature (C)'], aggfunc="mean"),
            cmap="coolwarm", annot=True, cbar=True);
plt.title("Average Temprature 2006-2016")
plt.plot();
```



Finally for visualization analysis we use a weather calendar. It makes the columns which are not familiar to us readable such as ‘Wind Speed (km/h)’. We showed 2008, Wind Speed.

```
#Drawing a heatmap
def facet_heatmap(data, color, **kws):
    values=data.columns.values[3]
    data = data.pivot(index='hour', columns='day', values=values)
    sns.heatmap(data, cmap='coolwarm', **kws)

#Joining heatmaps of every month in a year
def weather_calendar(year,weather): #Year= Any year in DataFrame. Weather=Any quantitative variable
    dfyear = df[df['year']==year][['month', 'day', 'hour', weather]]
    vmin=dfyear[weather].min()
    vmax=dfyear[weather].max()
    with sns.plotting_context(font_scale=12):
        g = sns.FacetGrid(dfyear,col="month", col_wrap=3) #One heatmap per month
        g = g.map_dataframe(facet_heatmap,vmin=vmin, vmax=vmax)
        g.set_axis_labels('Day', 'Hour')
        plt.subplots_adjust(top=0.9)
        g.fig.suptitle('%s Calendar. Year: %s.' %(weather, year), fontsize=18)
```



4. METHODOLOGY AND APPROACH

In a developing country and an economy like Hungary where major population is dependent on industry and tourism, weather conditions play an important and vital role in economic growth of the overall nation. So, weather prediction should be more exact and correct. Weather parameters are collected from the various stations of meteorological department in Szeged. This project uses a software called 'Kaggle Kernel' for programming. The programming language used is 'Python'. Fig. 1 visualizes the system in the form of a block diagram.

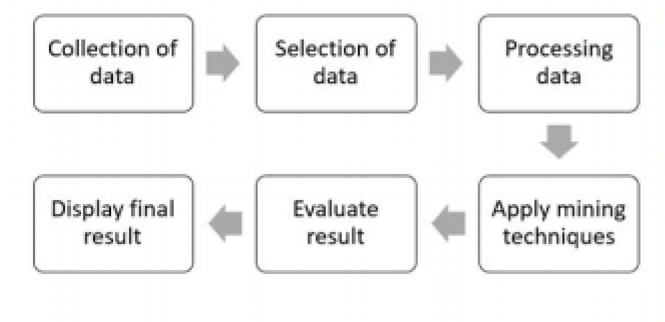


Fig. 1. System block diagram [2]

Three types of weather parameters are predicted: Temperature, Humidity, Daily Summary. Temperature is the measure of hotness or coldness, generally measured using thermometer. Units of temperature most frequently used are Celsius and Fahrenheit. Humidity is the quantity of water vapor present in the atmosphere. It is a relative quantity. Daily Summary is the brief of daily conditions such as "Light rain starting overnight." [1].

For the process of project, we used of a few standard and third party libraries. Data manipulation libraries are numpy and pandas. System library is glob. Plotting libraries are folium, seaborn, matplotlib, mpl_toolkits. Math operations library is math. Date manipulation library is datetime. Deep learning library is keras. Finally, for machine learning algorithms we used sklearn library.

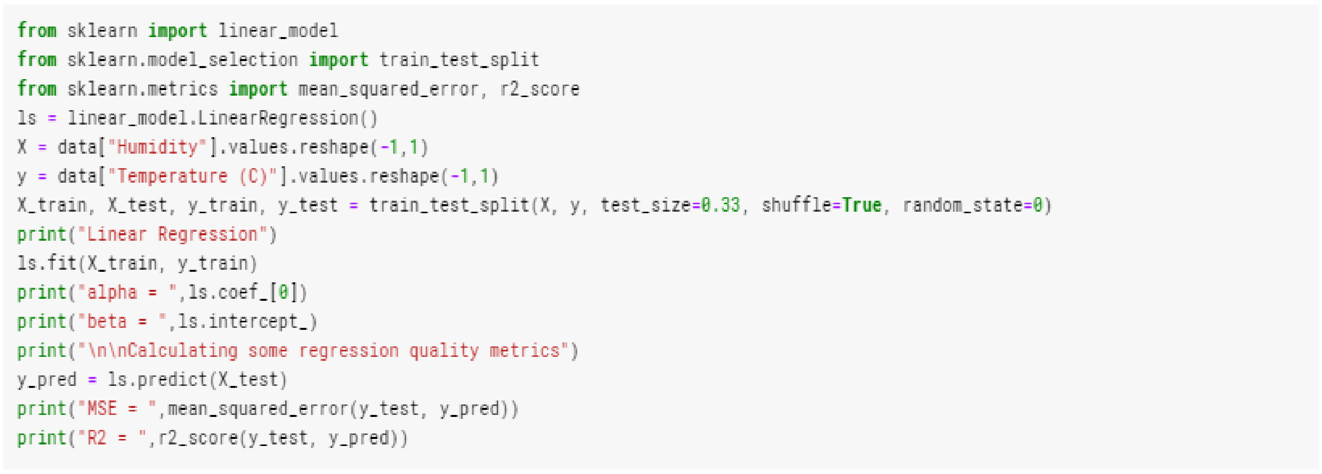
4.1. Linear Regression

It is a method used for defining the relation between a dependent variable (Y) and one or more independent variables or explanatory variables, denoted by (X). For multiple explanatory variable, the process is defined as Linear Regression (LR). The general equation for a linear regression is given as

$$\hat{y} = \beta0 + \beta1 * x1 + \beta2 * x2 + \ldots + \beta(p-n)\, x(p-n) + E$$

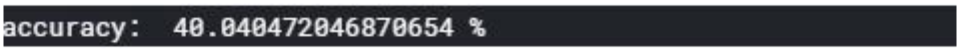
where ŷ is the predicted outcome variable -dependent variable- and xi are the predictor variables -independent variables- for i=1,2,..,n, β0 is the intercept or the value of ŷ when each xi equals zero, βi is the change in ŷ based on a one unit change in one of the corresponding xi, E is random error term associated with the difference between ŷj value and the actual yj value [1] .

Linear Regression is the most basic and frequently used predictive model for analysis. Regression estimates are generally used to describe the data and light relationship between one or more independent and dependent variables. Linear regression finds the best fit through the points, graphically. The best-fit line through the points is known as the regression line [2].



```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
ls = linear_model.LinearRegression()
X = data["Humidity"].values.reshape(-1,1)
y = data["Temperature (C)"].values.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, shuffle=True, random_state=0)
print("Linear Regression")
ls.fit(X_train, y_train)
print("alpha = ",ls.coef_[0])
print("beta = ",ls.intercept_)
print("\n\nCalculating some regression quality metrics")
y_pred = ls.predict(X_test)
print("MSE = ",mean_squared_error(y_test, y_pred))
print("R2 = ",r2_score(y_test, y_pred))
```

At this code, we are reshaping the array. After that we split arrays or matrices into random train and test subsets. These train subsets are started processing in ‘ls.fit (X_train, y_train). The mean squared error tells us how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the “errors”) and squaring them. The squaring is necessary to remove any negative signs. It’s called the mean squared error as we’re finding the average of a set of errors. According to formula, how close the predicted values are to the actual values, MSE will be smaller.



```
accuracy:  40.040472046870654 %
```

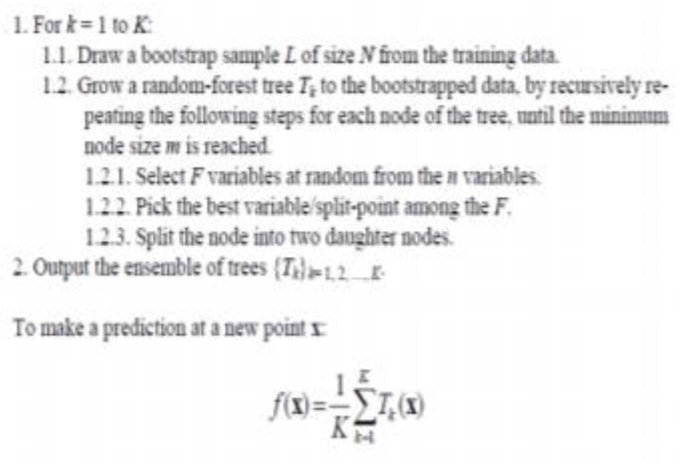
After the Linear Regression process, our accuracy score is 40.04%.

4.2. Random Forest Classifier

Random Forests (RF) is the most popular methods in data mining. First, Random Forest algorithm is a supervised classification algorithm.The method is widely used in different time series forecasting fields, such as biostatistics, climate monitoring, planning in energy industry and weather forecasting. Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object [3]. The difference between Random Forest algorithm and the decision tree algorithm is that in Random Forest, the process es of finding the root node and splitting the feature nodes will run randomly.

There are four advantages to use Random Forest algorithm. İt can be used for both classification and regression tasks. Overfitting is one critical problem that may make the results worse so Random Forest algorithm the classifier won't overfit the model when , if there are enough trees in the forest. The third advantage is the classifier of Random Forest can handle missing values, and the last advantage is that the Random Forest classifier can be modeled for categorical values [4].

There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage.



```
1. For $k = 1$ to $K$:
    1.1. Draw a bootstrap sample $L$ of size $N$ from the training data.
    1.2. Grow a random-forest tree $T_k$ to the bootstrapped data, by recursively re-
         peating the following steps for each node of the tree, until the minimum
         node size $m$ is reached.
        1.2.1. Select $F$ variables at random from the $n$ variables.
        1.2.2. Pick the best variable/split-point among the $F$.
        1.2.3. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_k\}_{k=1,2,\ldots,K}$.

To make a prediction at a new point $\mathbf{x}$:

    $f(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} T_k(\mathbf{x})$
```

Fig 2. Algorithm of RF for regression [5].

Where K represents the number of trees in the forest and F represents the number of input variables randomly chosen at each split respectively.

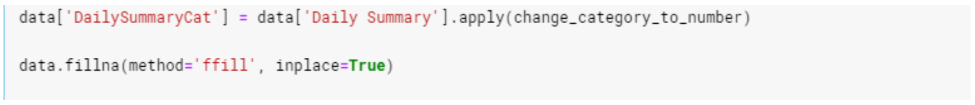
First of all we are going to change the categorical variables to numeric. Totally we have 64 Daily Summary categories, 27 Summary categories and 2 Precip Type categories. All category types different from each other and each one of them take different numeric value. Here we going to show you first 19 category of Daily Summary, first 18 category of Summary and all Precip Type categories.

```python
def change_category_to_number(DailySummaryCat):
    if DailySummaryCat=='Partly cloudy throughout the day.':
        return 1
    elif DailySummaryCat=='Mostly cloudy throughout the day.':
        return 2
    elif DailySummaryCat=='Foggy in the evening.':
        return 3
    elif DailySummaryCat=='Foggy overnight and breezy in the morning.':
        return 4
    elif DailySummaryCat=='Overcast throughout the day.':
        return 5
    elif DailySummaryCat=='Partly cloudy until night.':
        return 6
    elif DailySummaryCat=='Motly cloudy until night.':
        return 7
    elif DailySummaryCat=='Foggy starting overnight continuing until morning.':
        return 8
    elif DailySummaryCat=='Foggy in the morning.':
        return 9
    elif DailySummaryCat=='Partly cloudy until evening.':
        return 10
    elif DailySummaryCat=='Partly cloudy starting in the morning.':
        return 11
    elif DailySummaryCat=='Mostly cloudy starting overnight continuing until night.':
        return 12
    elif DailySummaryCat=='Partly cloudy starting in the afternoon.':
        return 13
    elif DailySummaryCat=='Partly cloudy starting overnight.':
        return 14
    elif DailySummaryCat=='Mostly cloudy starting overnight.':
        return 15
    elif DailySummaryCat=='Mostly cloudy until night and breezy in the afternoon.':
        return 16
    elif DailySummaryCat=='Mostly cloudy until evening.':
        return 17
    elif DailySummaryCat=='Foggy throughout the day.':
        return 18
    elif DailySummaryCat=='Partly cloudy starting in the morning.':
        return 19
```

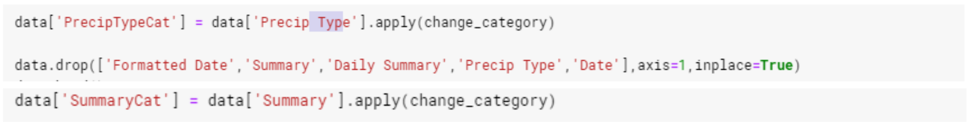
```python
def change_category(Summary):
    if Summary=='Partly Cloudy':
        return 1
    elif Summary=='Mostly Cloudy':
        return 2
    elif Summary=='Foggy':
        return 3
    elif Summary=='Clear':
        return 4
    elif Summary=='Overcast':
        return 5
    elif Summary=='Breezy and Overcast':
        return 6
    elif Summary=='Breezy and Partly Cloudy':
        return 7
    elif Summary=='Breezy and Mostly Cloudy':
        return 8
    elif Summary=='Dry and Partly Cloudy':
        return 9
    elif Summary=='Windy and Partly Cloudy':
        return 10
    elif Summary=='Light Rain':
        return 11
    elif Summary=='Breezy':
        return 12
    elif Summary=='Windy and Overcast':
        return 13
    elif Summary=='Humid and Mostly Cloudy':
        return 14
    elif Summary=='Drizzle':
        return 15
    elif Summary=='Windy and Mostly Cloudy':
        return 16
    elif Summary=='Breezy and Foggy':
        return 17
    elif Summary=='Dry':
        return 18
```

```python
def change_category(PrecipTypeCat):
    if PrecipTypeCat=='rain':
        return 1
    elif PrecipTypeCat=='snow':
        return 2
```

Assign a numerical value to the categorical field of class, by using the above function and made arrangements on dataset columns.



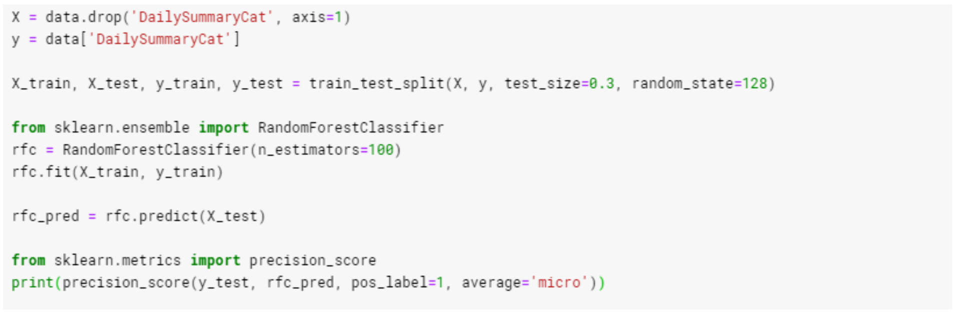
```
data['DailySummaryCat'] = data['Daily Summary'].apply(change_category_to_number)

data.fillna(method='ffill', inplace=True)
```



```
data['PrecipTypeCat'] = data['Precip Type'].apply(change_category)

data.drop(['Formatted Date','Summary','Daily Summary','Precip Type','Date'],axis=1,inplace=True)
```

```
data['SummaryCat'] = data['Summary'].apply(change_category)
```

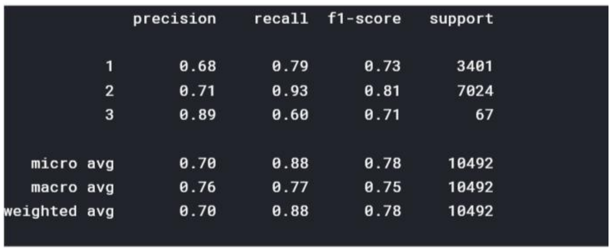
Daily Summary column data assigned to y variable. Rest of the columns assigned to x variable.



```
X = data.drop('DailySummaryCat', axis=1)
y = data['DailySummaryCat']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=128)

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)

rfc_pred = rfc.predict(X_test)

from sklearn.metrics import precision_score
print(precision_score(y_test, rfc_pred, pos_label=1, average='micro'))
```

Precision score is how accurate we predict from all of the classes. It should be as high as possible.

$$\textbf{\textit{Precision}} = \frac{TP}{TP + FP}$$

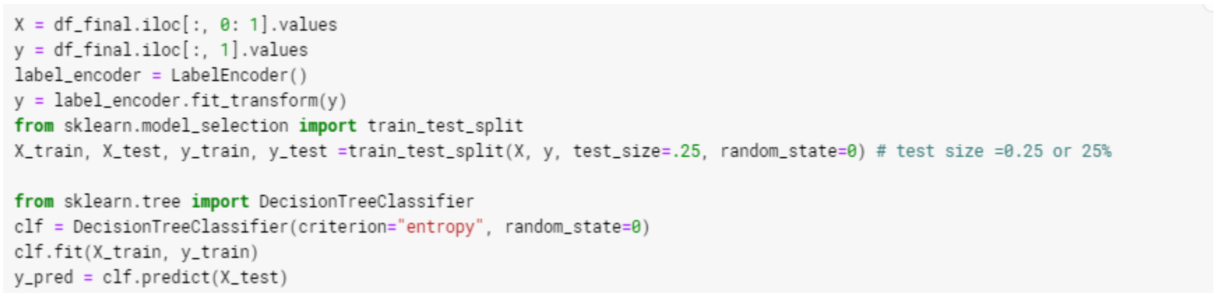


```
              precision    recall  f1-score   support

           1       0.68      0.79      0.73      3401
           2       0.71      0.93      0.81      7024
           3       0.89      0.60      0.71        67

   micro avg       0.70      0.88      0.78     10492
   macro avg       0.76      0.77      0.75     10492
weighted avg       0.70      0.88      0.78     10492
```

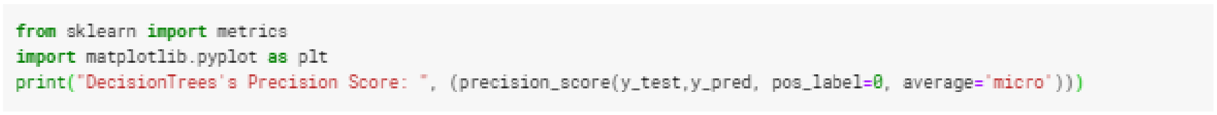
4.3. Decision Tree Classifier

Decision tree is an advanced knowledge discovery process with minimum time complexity and has an ease in the implementation. It establishes relationship between the various datasets by discovering the hidden patterns among the datasets which are huge and complex. Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data data features.

Decision tree constructs arrangement or relapse models as a tree structure. It separates a datasets into smaller and smaller subsets while in the meantime a related decision tree is incrementally created. The last result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node speaks to an order or choice. The highest decision node in a tree which relates to the best indicator called root node. Decision trees can dealwith both clear cut and numerical information [6].

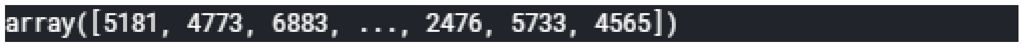


```
X = df_final.iloc[:, 0: 1].values
y = df_final.iloc[:, 1].values
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=.25, random_state=0) # test size =0.25 or 25%

from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion="entropy", random_state=0)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```



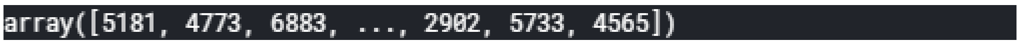
```
from sklearn import metrics
import matplotlib.pyplot as plt
print("DecisionTrees's Precision Score: ", (precision_score(y_test,y_pred, pos_label=0, average='micro')))
```

We tried to predict the weather Temperature (C) field. Like weather will be 5.181,7.5689. For this we used Decision Tree Classifier. (X) is feature matrix and (y)is target matrix. Firstly, we changed the categorical variables to numeric. According to our dataset and code y_pred is egual to



```
array([5181, 4773, 6883, ..., 2476, 5733, 4565])
```

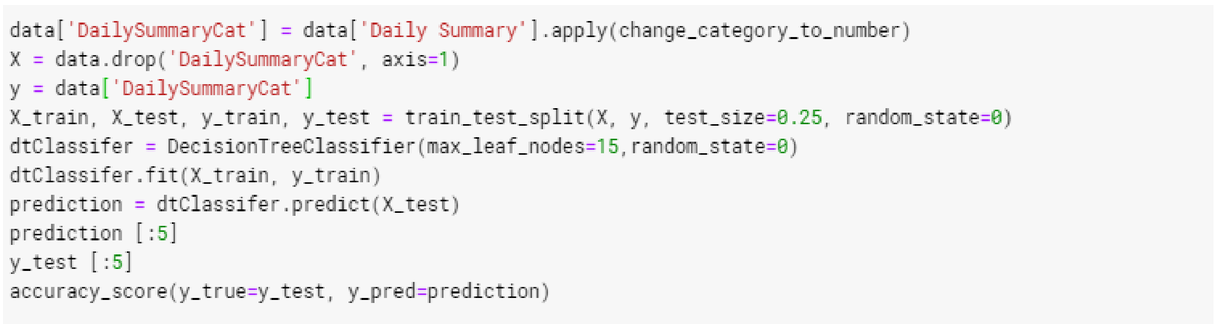
and y_test is equal to



```
array([5181, 4773, 6883, ..., 2902, 5733, 4565])
```

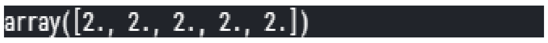
Reason for the difference is accuracy. This method’s accuracy is 54.44%. Also in this code, Decision Tree’s accuracy score is equal to precision score.

When we took Daily Summary field instead of Temperature (C) field. Our accuracy is going down rapidly. With this field our accuracy score is 29.74%.



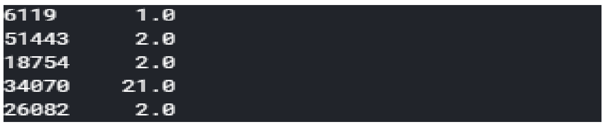
```
data['DailySummaryCat'] = data['Daily Summary'].apply(change_category_to_number)
X = data.drop('DailySummaryCat', axis=1)
y = data['DailySummaryCat']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
dtClassifer = DecisionTreeClassifier(max_leaf_nodes=15,random_state=0)
dtClassifer.fit(X_train, y_train)
prediction = dtClassifer.predict(X_test)
prediction [:5]
y_test [:5]
accuracy_score(y_true=y_test, y_pred=prediction)
```

Prediction [:5] is equal to



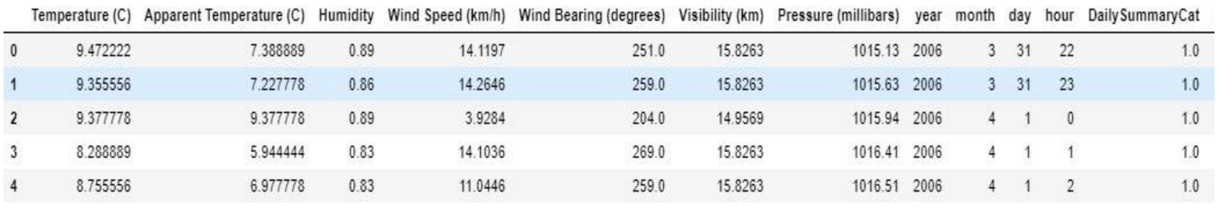
```
array([2., 2., 2., 2., 2.])
```

y_test [:5] is equal to



```
6119       1.0
51443      2.0
18754      2.0
34070     21.0
26082      2.0
```

When we changed 'Daily Summary' column values to object from int64, table changed.

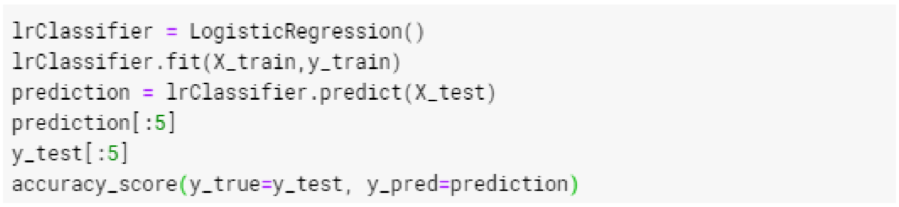


	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Pressure (millibars)	year	month	day	hour	DailySummaryCat
0	9.472222	7.388889	0.89	14.1197	251.0	15.8263	1015.13	2006	3	31	22	1.0
1	9.355556	7.227778	0.86	14.2646	259.0	15.8263	1015.63	2006	3	31	23	1.0
2	9.377778	9.377778	0.89	3.9284	204.0	14.9569	1015.94	2006	4	1	0	1.0
3	8.288889	5.944444	0.83	14.1036	269.0	15.8263	1016.41	2006	4	1	1	1.0
4	8.755556	6.977778	0.83	11.0446	259.0	15.8263	1016.51	2006	4	1	2	1.0

4.4. Logistic Regression

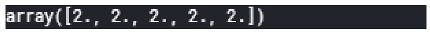
Regression analysis is a powerful statistical analysis technique. A dependent variable of our interest is used to predict the values of other independent variables in a data-set. We come across regression in an intuitive way all the time. Like predicting the weather using the data-set of the weather conditions in the past. It uses many techniques to analyse and predict the outcome, but the emphasis is mainly on relationship between dependent variable and one or more independent variable. Logistic regression analysis predicts the outcome in a binary variable which has only two possible outcomes.

In this project we use multinomial logistic regression. Multinomial Logistic Regression is the regression analysis to conduct when the dependent variable is nominal with more than two levels. Similar to multiple linear regression, the multinomial regression is a predictive analysis. Multinomial regression is used to explain the relationship between one nominal dependent variable and one or more independent variables [7].



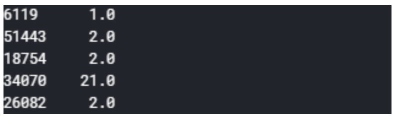
```
lrClassifier = LogisticRegression()
lrClassifier.fit(X_train,y_train)
prediction = lrClassifier.predict(X_test)
prediction[:5]
y_test[:5]
accuracy_score(y_true=y_test, y_pred=prediction)
```

When we used logistic regression our prediction is equal to



```
array([2., 2., 2., 2., 2.])
```

y_test is equal to



```
6119       1.0
51443      2.0
18754      2.0
34070     21.0
26082      2.0
```

In this method we also used Daily Summary field. Accuracy score is 29.74%. Because of the accuracy score we didn't continue with this method.

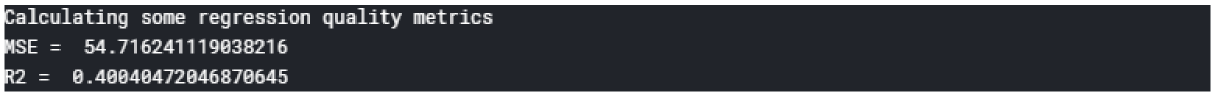
5. RESULT

The following is the experimental usage of the system proposed. The data is collected for every district for a period of 10 years from 2006 to 2016 for all the months and all of the days include with data for 24 different hours in a day. The training is done using the data from these years. There are 12 columns, Formatted Date, Summary, Precip Type, Temperature (C), Apparent Temperature (C), Humidity, Wind, peed (km/h), Wind Bearing (degrees), Visibility (km), Loud Cover, Pressure (millibars), Daily Summary.

We have varying range of results with respect to different input data and different classifiers. Prominently, two parameters for Linear Regression. Temperature (C) and Humidity are calculated through the hypothesis of the Linear Regression model. Secondly Daily Summary parameter used for Random Forest Classifier`s parameters. Temperature (C) parameter for Decision Tree and also Temperature (C) used for paramether to Logistic Regression. The accuracy score is also calculated for each of the algorithms. Accuracy scores are 40.04% for linear regression algorithm, 29.74% for Logistic Regression algorithm. Also 75.90% precision score for Random Forest Classifier algorithm and 54.44% for Decision Tree algorithm.

After implementing all, it is safe to say that Random Forest Classifier is best to use for this dataset among all the above methods used. Logistic Regration has %29,74 accuracy score because of that Logistic Regration is the worst one. Logistic Regression is a classification algorithm. It is widely used for classification problems. It requires large sample sizes because maximum likelihood estimates are less powerful at low sample sizes than ordinary least square. When consider those reasons, logistic regration is good choose for weather forecasting. But it used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. We used numerical values, Temperature (C) column`s values, for independent values. Also there is not any data predict a binary outcome in our dataset. Because of all of those reasons accuracy score is worst one in four methods [8].

After our research, we have seen that they use Linear Regression in forecasting. We got %40,04 accuracy scores for linear regression. In our data set the relationship between the variables is linear and predictors uncorrelated. Result is little bir more improved version of Logistic regression but it is not enough for correct predictions. Here we can see at r2_score is %40,04 to but mean_squared_error is 54.71 and this rate much more than expected.



```
Calculating some regression quality metrics
MSE =  54.716241119038216
R2 =  0.40040472046870645
```

We assume that classification methods appropriate for our dataset. Decision Tree method increased success rate. Nearly half of the predictions gave true value but but it is not enough for correct predictions applied methods befour.

Random Forest Classifier gave the best results in other methods. Random Forest Classifiers facilitate the reduction in the over-fitting of the model and these classifiers are more accurate than the decision trees in several cases. But random forests exhibit real-time prediction but that is slow in nature. They are also difficult to implement and have a complex algorithm.

6. CONCLUSION

In this project, we used supervised learning methods which are Linear Regression, Logistic Regression, Decision Tree and Random Forest classification. Random Forest gives the best result with 75.90%. Also Random Forest is really time efficient and easy to perform with right feature models. Decision Tree and Linear Regression are not good as Random Forest but they have capability for using. Logistic Regression does not fit to our problem for real. It gives bumpy results. We also tried to use Support Vector Machine (SVM) but we failed. SVM is not good enough for our task and slower. The hardest part is trying some ML algorithms, because there are many variations and trying each algorithms takes too long. Random Forest works faster and gives good results at the end. Making prediction for unknown future is tricky and hard. This project our first machine learning experience and according to everything which is learnt and searched by us weather forecasting is hard to perform task, if we keep trying, we can get much better results.

REFERENCES

[1] S. Prabakaran, P. Naveen Kumar and P. Sai Mani Tarun "RAINFALL PREDICTION USING MODIFIED LINEAR REGRESSION" Department of Computer Science and Engineering, SRM University, Kattankulathur, Chennai, India, ARPN Journal of Engineering and Applied Sciences. Vol. 12 (12), 2017, 3715-3718.

[2] S. Gupta, K. Indumathy, G. Singhal "Weather Prediction Using Normal Equation Method and Linear regression Techniques" Department of Electronics & Communication Engineering, VIT University, International Journal of Computer Science and Information Technologies, Vol. 7 (3) , 2016, 1490-1493.

[3] Patel, S.(2017, 18 May). " Random Forest Classifier." https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1

[4] How Random Forest Algorithm Works in Machine Learning. (2017, 24 Oct). https://syncedreview.com/2017/10/24/how-random-forest-algorithm-works-in-machine-learning/

[5] Htike, Z. , Naing W. "FORECASTING OF MONTHLY TEMPERATURE VARIATIONS USING RANDOM FORESTS" , Department of Mechatronics Engineering, Faculty of Engineering, International Islamic University Malaysia, Malaysia ,ARPN Journal of Engineering and Applied Sciences. Vol. 10, No 21, November, 2015 .

[6] Narasimha Prasad L. Naidu M. "An Efficient Decision Tree Classifier to Predict Precipitation Using Gain Ratio". The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3 (3), 2013, 674-675.

[7] Srivastava N. "A logistic regression model for predicting the occurrence of intense geomagnetic storms ". Annales Geophysicae, 23, 2005, 2970–2971.

[8] Brid S. (2018, Oct 17). "Brief on Regression Analysis" https://medium.com/greyatom/logistic-regression-89e496433063

[9] Ashwini M. Jadhawar B.A. "Weather forecast prediction: a Data Mining application" International Journal of Engineering Research and General Science Vol.3 (2), 2015, 1280-1282

[10] Paras M. Sanjay M. "A Simple Weather Forecasting Model Using Mathematical Regression" Department of Electronics & Communication Engineering, College of Technology, G.B. Pant University of Agriculture & Technology, 2012, 163-165.