Dated : 16/10/2022

A

Project Report

on

Rainfall-Prediction-Weather Forecasting

Machine Learning Project

Weather has been an important field of study for human being from the ancient time. Many countries' economies are more or less dependent on agriculture and weather has huge impact on the same. Hence the weather has substantial impact on the world economy. The weather can make or destroy many things.

One can not change the weather but if weather is forecasted, the bad impact of major events of weather can be avoided or reduced by being more prepared for the upcoming situation(s).

Initially weather forecasting was done by human beings by manual calculations on the basis of data of recorded weather events. There were many drawbacks of this approach and those are:

1. It was a long process
2. Accuracy was not enough
3. Storing a huge amount of data was not possible
4. It could be done for few locations only

Machine Learning has made it fast, accurate, simple and reliable by providing us better and more accurate results for multiple locations, days and so on.

Let's have a look on the practical problem solved by Machine Learning.

1. **Problem Definition:** We have a data set of daily weather events (quantitative data of current state of atmosphere) of different locations of Australia for 10 years.

We have two targets

a) Whether it will rain tomorrow at a particular location

b) How much rainfall in mm is expected to be there on the very next day ?

Based on the learning of a selected model we can get good accuracy in the above prediction.

2. **Data Analysis:** First of all the data is loaded in the memory by using pandas read_csv() method in a data set called ds. Following to this a data frame is made using pd's DataFrame() method , the name of the data frame is df.

```
ds = pd.read_csv('weatherAUS.csv')

df = pd.DataFrame(data=ds)

print(df)
```

As the collection of data happens at a very root level and there is always a probability of getting missing, improper, impure, redundant, unwanted and noisy data and we can not train a machine learning model on such data. If we use such sporadic data it will spoil the whole function and may lead us to a set of wrong or incorrect predictions.

Hence all the above-mentioned impurities or anomalies should be removed before we train our model on it. So now it is time for observing the values of the Data Frame and then it is checked for null values by using isna() or isnull() function of dataframe.

```
df.isna().sum()
```

There are so many null values in the given dataframe.

If null value, space or any other unwanted characters are available in the data frame then these irregularities will impact the model's prediction badly. So, we need to remove these. For filling the NaN, space or " " blank values we use following commands:

```
for i in df.columns:
    if(df.dtypes[i]=='object'):
        df[i].fillna(df[i].mode(),inplace=True)
    else:
        df[i].fillna(df[i].mean(),inplace=True)
```

Here we fill the null values of string columns with mode of the column and values of numeric columns with mean of the column.

There are still some nan and space values in the data frame so we remove these by the following commands:

```
for i in df.columns:
    mode = df[df[i]!=" "][i].mode()[0]
    df[i] = df[i].replace(" ",mode)
    df[i] = df[i].replace("NaN",mode)
```

df[i] = df[i].replace(np.nan,mode)

Now the data frame is completely treated for all the missing values.

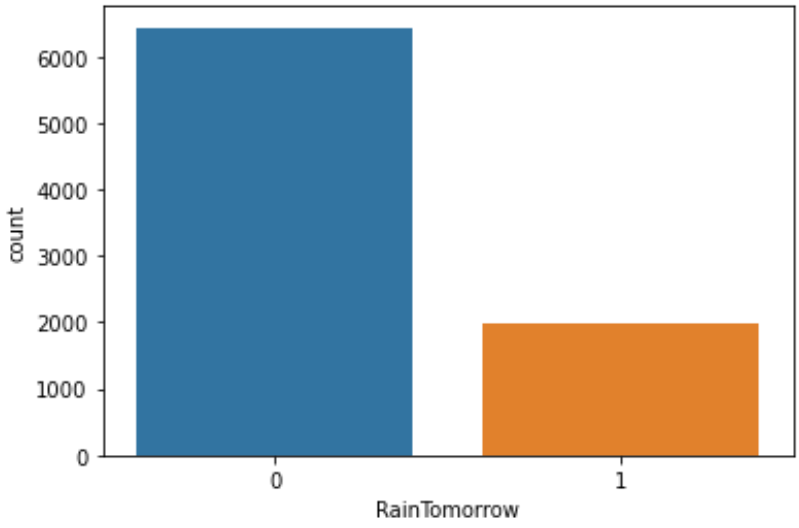
The data frame has total 23 features and 7 features are string (object) type and other 16 features are float64 type. We need to check the cardinality of the features so as to understand that how many categories we have in a particular features. Our target feature RainTomorrow has two unique categories that means targe has two unique values across the data frame so its cardinality is 2.

The features with high cardinality may affect the performance of the model so we should drop such features with high cardinality lets say 20 or above.

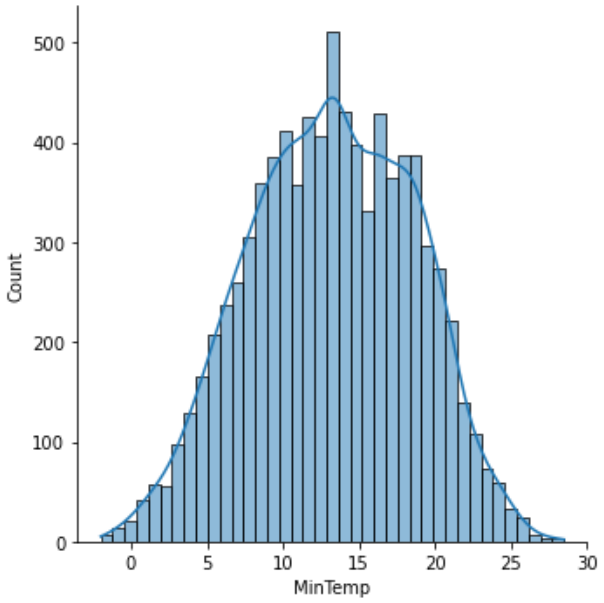
Since Machine understand only numbers so we need to convert the string features with numeric values and for the same we use any encoder like we use fit_transform method of LabelEncoder.

After performing the above operation our data is ready to be used for training of a Machine Learning algorithm. But before doing so we need to get insight of the data by few useful visualization methods. This is done in the next step of EDA or Exploratory Data Analysis.

3. **EDA Concluding Remark:** First of all in this process we get to know the counting of occurrence of different classes of the target variable RainTomorrow. And we found that class 0 (N) is occurring over 6000 times and class 1(Y) is occurring nearly 2000 times. This visualization clearly depict that there is class imbalance and it will definitely affect the performance of the Machine Learning Model. So this imbalance need to be removed by oversampling or under sampling and it will be done in Pre-Processing stage.



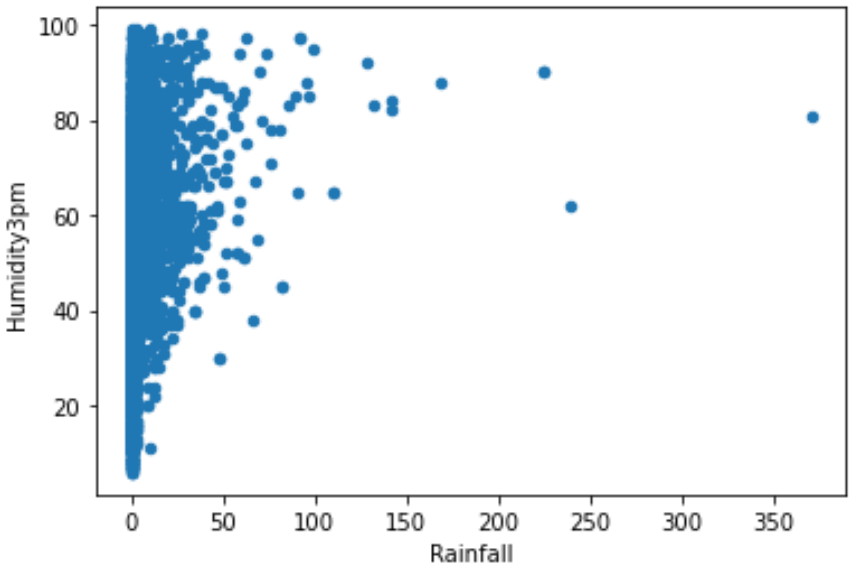
Now we check the distribution of the data for each column or features by using distribution plot.

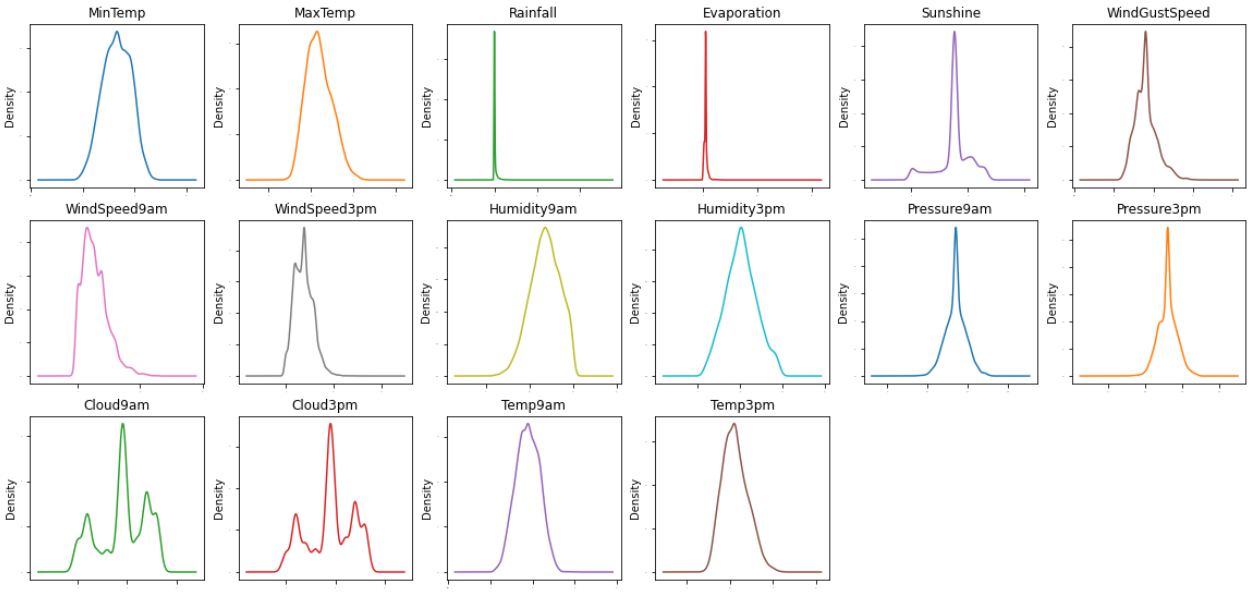


And we get to know that many features are not normally distributed and that need to be treated.

Then we check the corelation of all the features with target feature using another visualization method heatmap. It gives us a clear idea of how much all the independent variable contribute to the target feature. If any feature is contributing very less or two independent features are interdependent on each other then it will affect the performance of the model. Here at this juncture we can decide to drop such features for better results or prediction.

We use few more visualization methods like scatterplot, line plot , Box Plot , Area Plots and many more to get insight of the data. Like Box Plot can tell us whether any value(s) of the different columns fall under outliers or not.





We do Univariate Analysis, Byvariate Analysis and Multivariate Analysis to know the insight of the data before we actually train the Machine.

We further use df.info() and df.describe() methods to get further specific information about data frame features and data.

4. **Pre-Processing Pipeline:** In this stage we check the features for outliers by using any of the methods used for the same. Like we can use the Zscore method to get to know whether or not the values of different columns of the dataset lies as an outlier. If so we calculate the data loss which may be caused due to the outlier removal and if the data loss is under affordable lime i.e. less than 10%, we remove the outliers by using following commands.

```
from scipy.stats import zscore

import numpy as np

z=np.abs(zscore(df))

np.where(z>3)

df_new_z=df[(z<3).all(axis=1)]

df_new_z
```

In this prediction model the data loss is approximately 5% and that can be afforded as it is less than threshold limit. So we remove the outliers and use the new data frame which is created after removing the outliers.

For moving ahead the data frame is divided in feature (x) and target (y) variables by the following commands:

```
x=df.drop('RainTomorrow', axis=1)

y=df['RainTomorrow']
```

Before the further processing VIF value of each column is calculated by a function and it is found that two features Pressure9am and Pressure3pm have higher VIF values. It clearly shows that these features have multicollinearity and that has to be removed to improve the performance of the model to be trained.

Here feature Pressure3pm is contributing less to the target features hence the feature is removed.

Now we check the skewness present in the dataframe by using df.skew() method and using the sort_values() we display the skewness in ascending order just to get an idea that how much a feature is skewed in an orderly manner.

In this data frame the skewness is present in few features and many features have skewness at an acceptable level (-0.5 to +0.5). This skewness need to be treated. For treating the skewness we can use any of the transformation method like log transformation , sqrt transformation or box-cox transformation. We can also use power_transform from sklearn.preprocessing library or any other ready to use transformer. After transforming the data we get to know by using the skew() method that the skewness of the features of the dataset is at acceptable level. Now we can pre-process the data for further few things.

Its now time to scale the data and we can use any of the available scaler like MinMaxScaler or StandardScaler. Since outlier are already removed so it is better to use MinMaxScaler otherwise we could use StandardScaler as the performance of MinMaxScaler is affected by outliers.

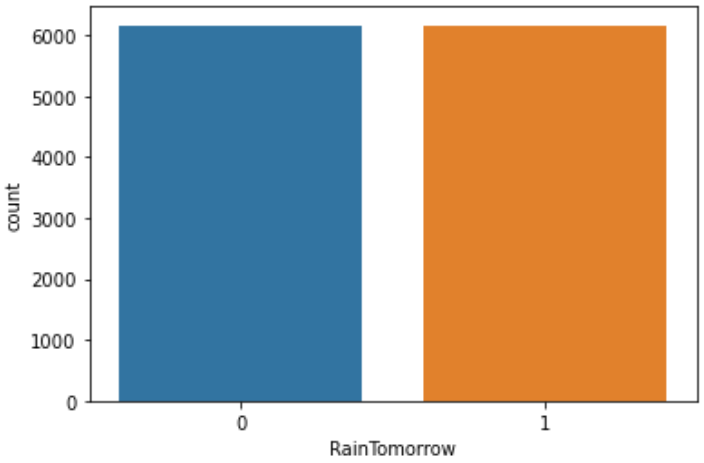
Here in this dataframe we use MinMaxScaler since outliers are already remove to get good performance.

```
from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler()

x = scale.fit_transform(x)
```

Finally we need to balance the class of the target variable by sung oversampling or under sampling.

Here in this example we use oversampling by SMOTE by using the following code.

```
from imblearn.over_sampling import SMOTE

smt=SMOTE()

x,y=smt.fit_resample(x,y)
```



Now all the pre processing operation are done and our data frame is ready to be used for training a model.

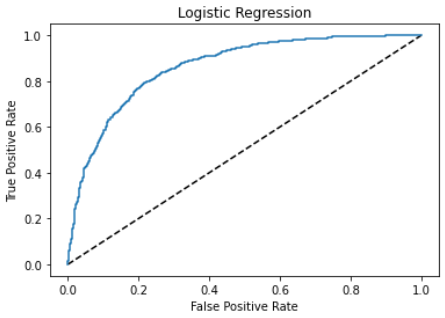
5. Building Machine Learning Models: In this step first of all we try to find out the best random state by the following code:

```
tr_acc=0

tr_i=0

for i in range(0,50):

    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=i)

    lr.fit(x_train,y_train)

    pred_test=lr.predict(x_test)


    if(round(accuracy_score(y_test,pred_test)*100)>tr_acc):

        tr_acc=accuracy_score(y_test,pred_test)*100

        tr_i=i

print(" Best Test Accuracy ",tr_acc," at ",tr_i)
```

And it is found that 47 is best random state. Now the data is divided in train and test variable here we take 20% data for testing and 80% keep for training.

Now we try different models for their accuracy on the data. First we try LogisticRegression by creating its instance and then training the model by fit() method and predicting the values by predict() method and then check the accuracy of the model. We get accuracy_score , draw confusion matrix , print classification report and finally check the cross validation score of each model.

We generate AUC-ROC curve to get an idea of the effectiveness of the model.



After testing few models it is found that DecisionTreeClassifer is best model out of the following model.

LogisticRegression

KNN Classifier

SVM Classifier

RandomeForestClassifier

DecisionTreeClassifer

So now it is time to find out the best parameters of the best algorithms / model to make its prediction better using Hyper Parameter Tuning and we use GridSearchCV for the same as per the following code :

```
parameters={'criterion':['gini', 'entropy', 'log_loss'],'max_features':['auto','sqrt', 'log2'],'random_state':[10,30,50],'max_depth':[1,2,3,4]}

clf = GridSearchCV(dtc,parameters,cv=39)

clf.fit(x_train,y_train)

print(clf.best_params_)
```

The model is finally trained with the best parameters as:

```
dtc=DecisionTreeClassifier(criterion='entropy', max_depth=4, max_features='auto', random_state=50)

dtc.fit(x_train,y_train)

dtc_score_training = dtc.score(x_train,y_train)

pred = dtc.predict(x_test)
```

```
print('accuracy score =',accuracy_score(y_test,pred))

print(confusion_matrix(y_test,pred))

print('classification_report =',classification_report(y_test,pred))
```

The accuracy of the model is approximately 75% which is at acceptable level. The model is selected on the basis of minimum difference between the accuracy score and cross validation score.

6. Concluding Remarks: When the best model is trained, it should be saved for future use. So we save the model by using pickle.

```
import pickle

outfile=open('a1.pkl','wb')

pickle.dump(rfc,outfile)

outfile.close()
```

The model can be retrieved by:

```
infile=open('a1.pkl','rb')

new_rf=pickle.load(infile)

infile.close()
```

In the second task we need to predict how much rainfall can happen tomorrow. It's a regression analysis. In this we perform all the operation till pre processing almost in the same manner with respect to the Rainfall feature.

The data is then broken into train and test with same 80 : 20 ratio after getting best random state.

Then few model for regression are trained and tested for the accuracy and the models are:

LinearRegression

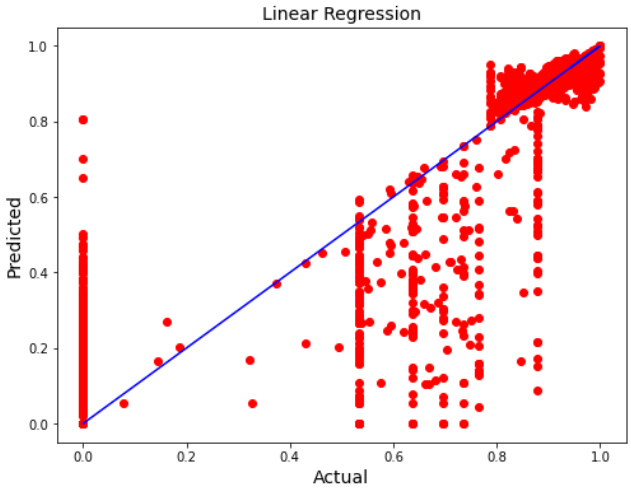
Ridge

SVR

RandomForestRegressor (Ensemble)

BaggingRegressor

It is further found that LinearRegression has best performance.



After doing hyperparameter tuning we get best parameters and the best model is finally trained with best parameters using fit() method and prediction is made using predict() method.

Similar to the classification the best model is saved using pickle and later can be retrieved by pickle.

Finally a data frame is prepared and displayed for comparison of actual values and predicted values using the following code.

pred_decision=y_test

conclusion=pd.DataFrame([new_rf.predict(x_test)[:],pred_decision[:]],index=["Predicted","Original"])

conclusion

In classification best model is DecisionTreeClassifer and in Regression best model is LinearRegression.

There is no thumb rule to say that which model is best for which data so we need make best use of visualization , pre processing , PCA, SMOTE, AUC-ROC, Ensemble , Scaling, Regularization etc methods to get best prediction from tested models and should use the model as best which gives us best accuracy and best cross validation score with minimum difference between them. The more the difference between these two the more the chances of overfitting.

Hence keeping the above facts in mind best model should be choosen.

Submitted by : Amit Puri Goswami