Rainfall Prediction - Weather Forecasting

Introduction:

Prediction in the field of meteorological phenomenon is complex due to the high number of variables on which it depends.

In this article, I will be implementing a predictive model on Rain dataset to predict whether or not it will rain tomorrow in Australia. The dataset contains about 10 years of daily weather observations of different locations in Australia. Rain tomorrow is the target variable to predict. It means did it rain the next day, Yes or No? This column is Yes if the rain of that day was 1mm or more.

1-Problem Statement/Problem Definition:

Can we predict whether it will rain tomorrow or not using data?

Solution: Classification model (Random Forest Classifier) using Machine Learning can be used for forecasting whether it will rain tomorrow or not.

Dataset: This dataset contains about 10 years of daily weather observations. In this article, I'll be briefly explaining the different sections.

Design a predictive model with the use of Machine Learning algorithms to forecast weather or not it will be rain tomorrow in Australia.

2-Dataset Description (Data Analysis):

Number of Columns:23

Number of Rows:8425

This dataset consists of 8425 rows and 22 columns with Rain Tomorrow being the dependent variables. The dataset has Float and Object values for various variables. The dataset can be loaded using a method read_csv (). The shape property is used to find the dimensions of the dataset.

Data source-Weather

In [2]:
```
df=pd.read_csv("Weather.csv")
df.head()
```

Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	100
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	101
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	100
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	101
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	101

5 rows × 23 columns

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
       'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

In [4]:
```
df.shape
```

Out[4]:
```
(8425, 23)
```

3-Data Pre-processing:

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8425 entries, 0 to 8424
Data columns (total 23 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           8425 non-null   object 
 1   Location       8425 non-null   object 
 2   MinTemp        8350 non-null   float64
 3   MaxTemp        8365 non-null   float64
 4   Rainfall       8185 non-null   float64
 5   Evaporation    4913 non-null   float64
 6   Sunshine       4431 non-null   float64
 7   WindGustDir    7434 non-null   object 
 8   WindGustSpeed  7434 non-null   float64
 9   WindDir9am     7596 non-null   object 
 10  WindDir3pm     8117 non-null   object 
 11  WindSpeed9am   8349 non-null   float64
 12  WindSpeed3pm   8318 non-null   float64
 13  Humidity9am    8366 non-null   float64
 14  Humidity3pm    8323 non-null   float64
 15  Pressure9am    7116 non-null   float64
 16  Pressure3pm    7113 non-null   float64
 17  Cloud9am       6004 non-null   float64
 18  Cloud3pm       5970 non-null   float64
 19  Temp9am        8369 non-null   float64
 20  Temp3pm        8329 non-null   float64
 21  RainToday      8185 non-null   object 
 22  RainTomorrow   8186 non-null   object 
dtypes: float64(16), object(7)
memory usage: 1.5+ MB
```

- Dataset has two data types: float64, object
- Except for the Date, Location columns every column has missing values.

Let’s generate descriptive statistics for the dataset using the function describe () in pandas.

4-Finding Categorical and Numerical Features in a Data set:

#Categorical & Numerical features in Dataset:

List of categorical & Numerical columns

In [11]:
```
numCol=[]
catCol=[]

for col in df.columns:
    if df[col].dtype=='O':
        catCol.append(col)
    else:
            numCol.append(col)
```

In [12]:
```
print("List of categorical columns:",catCol)
print("List of numerical columns:",numCol)
```

```
List of categorical columns: ['RainToday', 'RainTomorrow']
List of numerical columns: ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9
am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'TempDiff', 'HumidityDiff', 'CloudDiff', 'WindSpeedD
iff', 'PressureDiff']
```

5-Missing Value Imputation:

There are different ways of handling missing values in the data. We can delete those observations or can fill them with statistical measures. In this case, statistical measures like mode and mean have been used to replace missing values in categorical and numerical variables, respectively.

Machine learning algorithms can’t handle missing values and cause problems. So, they need to be addressed in the first place. There are many techniques to identify and impute missing values.

If a dataset contains missing values and loaded using pandas, then missing values get replaced with NaN (Not a Number) values. These NaN values can be identified using methods like ***isna ()*** or ***isnull ()*** and they can be imputed using ***fillna ().*** This process is known as **Missing Data Imputation**.

```
df.isnull().sum()
```

```
Date                 0
Location             0
MinTemp             75
MaxTemp             60
Rainfall           240
Evaporation       3512
Sunshine          3994
WindGustDir        991
WindGustSpeed      991
WindDir9am         829
WindDir3pm         308
WindSpeed9am        76
WindSpeed3pm       107
Humidity9am         59
Humidity3pm        102
Pressure9am       1309
Pressure3pm       1312
Cloud9am          2421
Cloud3pm          2455
Temp9am             56
Temp3pm             96
RainToday          240
RainTomorrow       239
dtype: int64
```

6-Exploratory Data Analysis (EDA):

1-Descriptive Statistics

Descriptive Statistics

In [18]:
```
df.describe(include='all')
```

Out[18]:

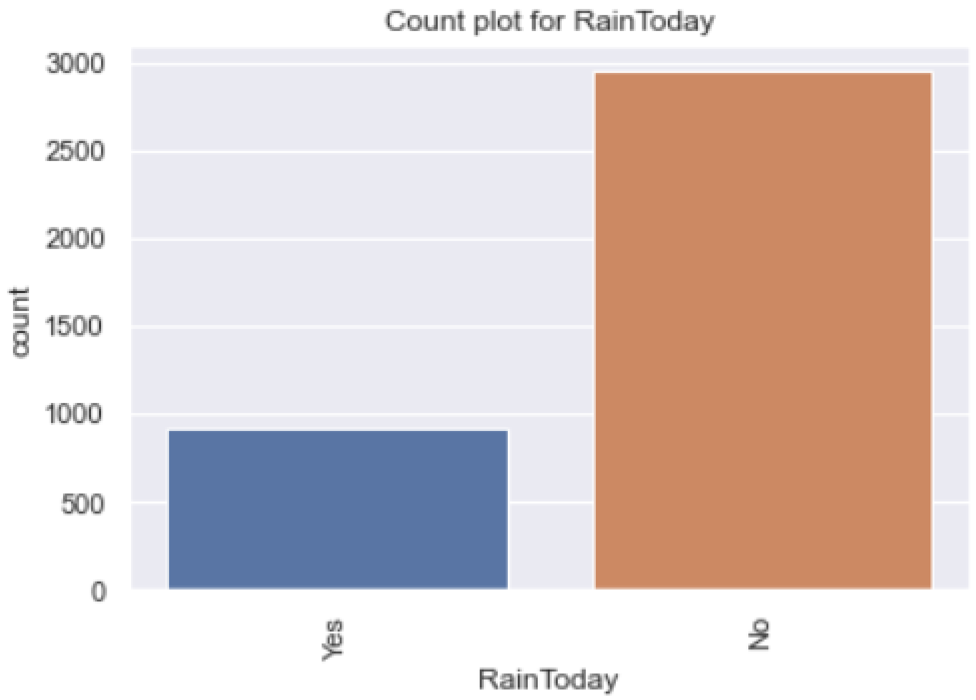
	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
count	3863.000000	3863.000000	3863.000000	3863.000000	3863.000000	3863.000000	3863.000000	3863.000000	3863.000000	3863.000000
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	14.006549	24.668651	2.477505	5.249003	7.818121	40.946932	16.095004	19.916645	63.967124	49.612218
std	5.196622	6.068042	7.782405	3.648030	3.826440	13.894644	9.654440	8.803734	16.341377	17.710319
min	-0.700000	10.800000	0.000000	0.000000	0.000000	13.000000	0.000000	0.000000	11.000000	6.000000
25%	10.100000	20.200000	0.000000	2.600000	5.200000	31.000000	9.000000	13.000000	53.000000	38.000000
50%	13.900000	24.000000	0.000000	4.600000	8.800000	39.000000	15.000000	20.000000	64.000000	49.000000
75%	17.900000	29.100000	0.800000	7.000000	10.800000	50.000000	22.000000	26.000000	75.000000	61.000000
max	28.500000	43.600000	168.400000	37.000000	13.900000	102.000000	61.000000	52.000000	99.000000	98.000000

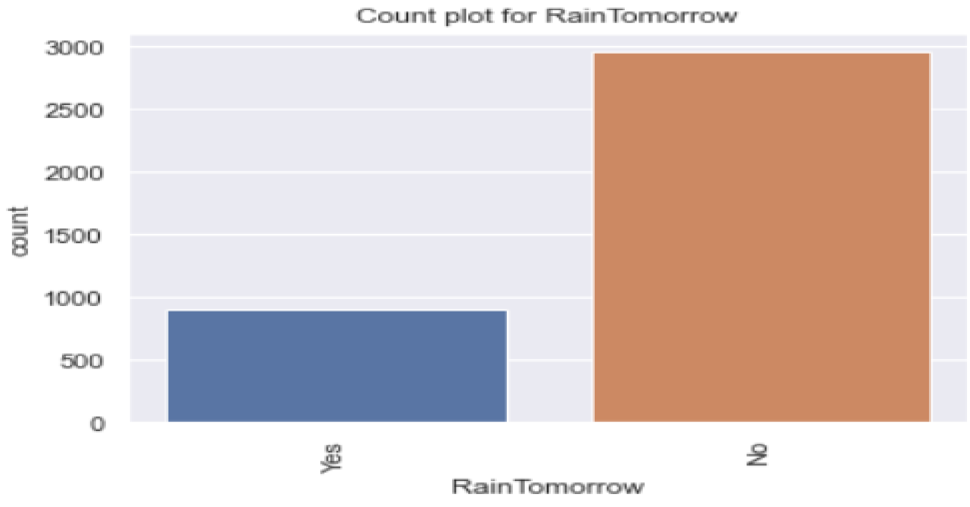
- MinTemp ranges from -0.70 to 28.50 with a standard deviation of 5.19
- Hottest day in Australia had 43.60 degrees
- On average Wind speed remains pretty similar at 9 am and 3 pm.

More insights can be derived from descriptive statistics. The idea is to get a feel of the data and later on depending on requirements; different parameters can be assessed.

2-Univariate Analysis

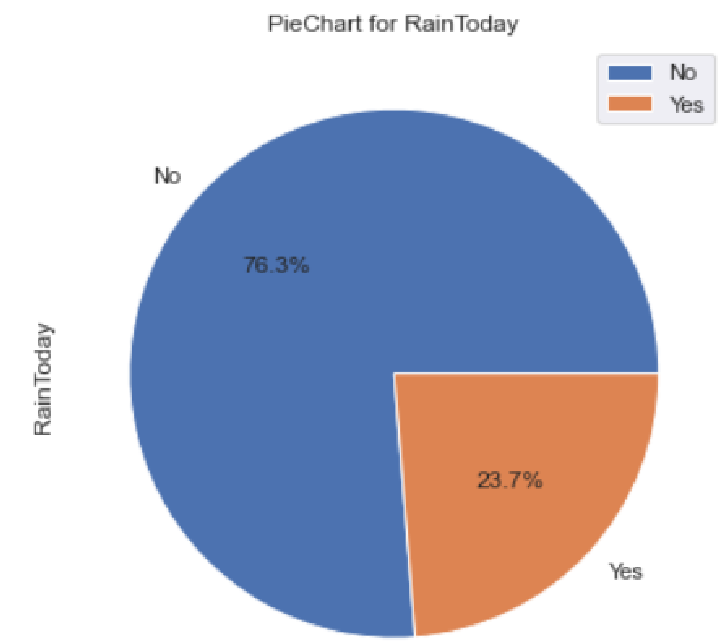
```
for i in catCol:
    plt.figure()
    sns.set_theme(style="darkgrid")
    sns.countplot(df[i])
    plt.xticks(rotation=90)
    plt.title(f"Count plot for {i}")
    plt.plot()
    plt.show()
```

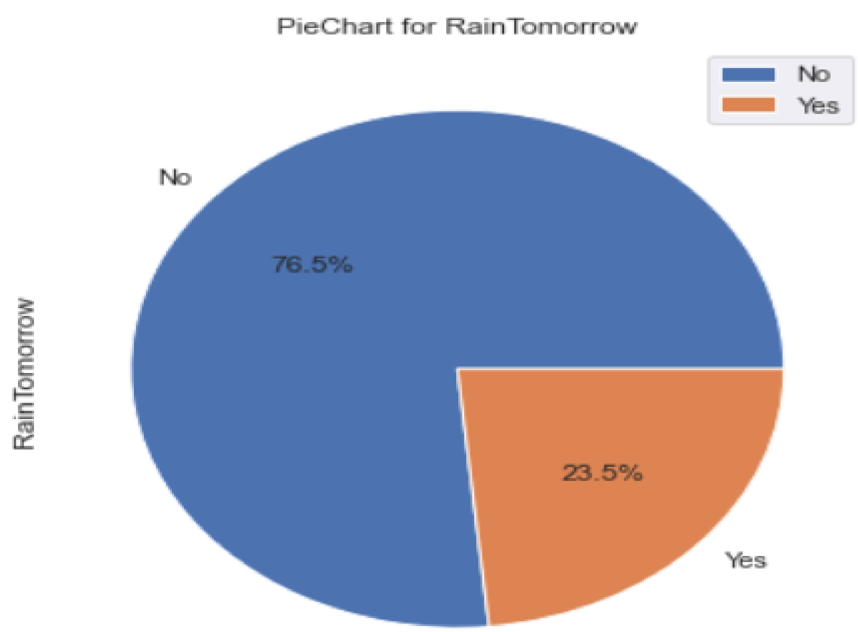




```
for i in catCol:
    plt.figure()
    sns.set_theme(style="darkgrid")
    countsDF= pd.DataFrame(df[i].value_counts())
    plot= countsDF.plot.pie(subplots=True,autopct="%.1f%%",figsize=(11,6))
    plt.title(f"PieChart for {i}")
    plt.plot()
    plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```





Looks like the Target variable is imbalanced. It has more 'No' values. If data is imbalanced, then it might decrease the performance of the model. As this data is released by the meteorological department of Australia, it doesn't make any sense when we try to balance the target variable, because the truthfulness of data might decrease. So, let me keep it as it is.

7- Outliers detection and treatment:

What is an outlier?

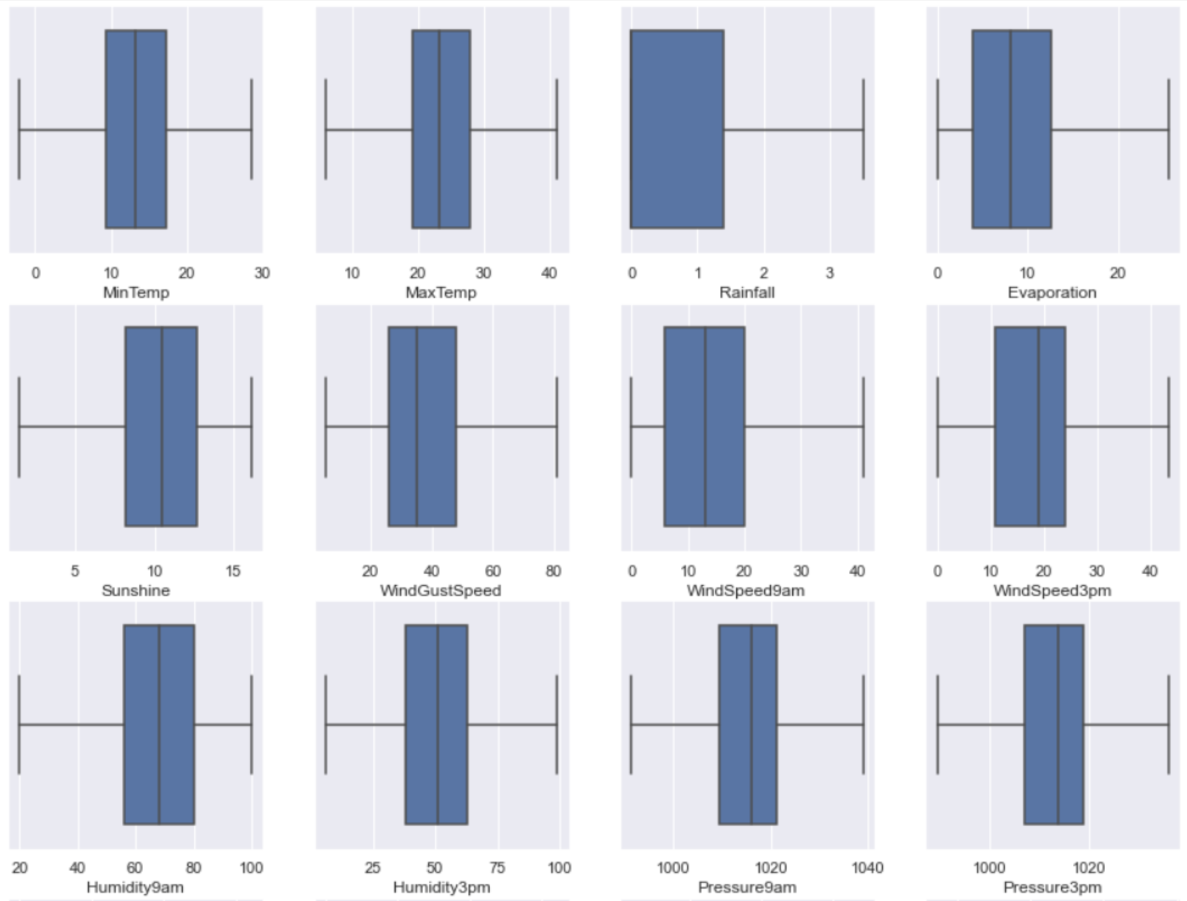
An Outlier is an observation that lies an abnormal distance from other values in a given sample. They can be detected using visualization (like boxplots, scatter plots), Z-score, statistical and probabilistic algorithms, etc.

Outlier Treatment to remove outliers from Numerical Features:

```
lsUpper = []
lsLower = []
def removeOutliers(numerical):
    for i in range(len(numerical)):
        q1 = df[numerical[i]].quantile(0.25)
        q3 = df[numerical[i]].quantile(0.75)
        IQR = q3-q1
        minimum = q1 - 1.5 * IQR
        maximum = q3 + 1.5 * IQR
        df.loc[(df[numerical[i]] <= minimum), numerical[i]] = minimum
        df.loc[(df[numerical[i]] >= maximum), numerical[i]] = maximum
removeOutliers(numerical)
```

```
num_of_rows = 4
num_of_cols = 4
fig, ax = plt.subplots(num_of_rows, num_of_cols, figsize=(15,15))
print(numerical)
i=0;j=0;k=0;
while i<num_of_rows:
    while j<num_of_cols:
        sns.boxplot(df[numerical[k]], ax=ax[i, j])
        k+=1;j+=1
    j=0;i+=1
plt.savefig('after_removing_outliers_from_numerical_columns.png')
plt.show()
```

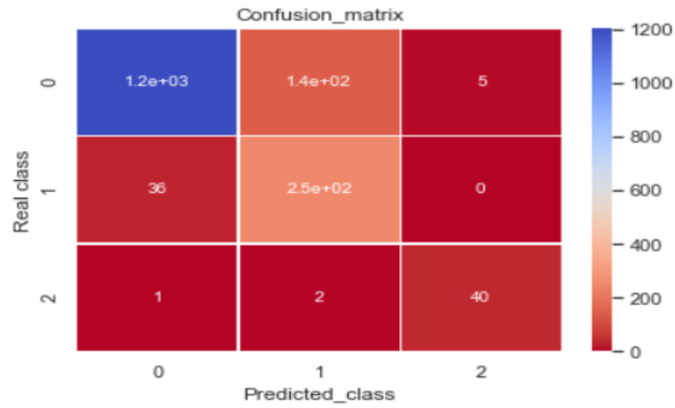
```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',
       'Temp9am', 'Temp3pm'],
      dtype='object')
```



8-Correlation Between Variables:

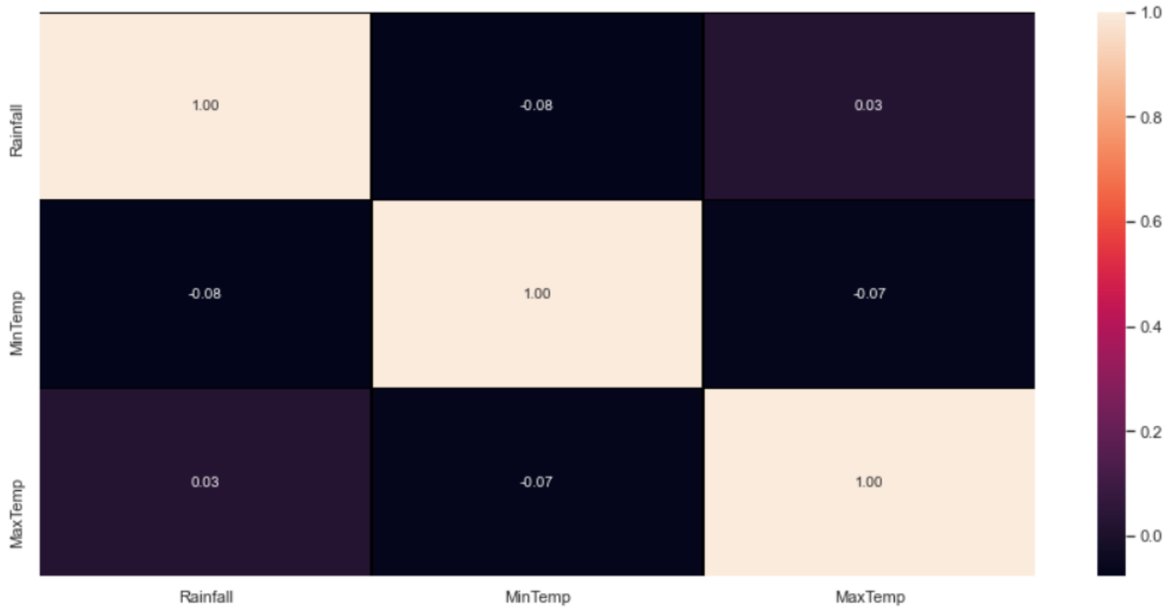
Correlation helps us to find how independent variables are affecting the dependent variables and also at the same time helps us to remove the variables which are highly correlated to each other.

Correlation is a statistic that helps to measure the strength of the relationship between two features. It is used in bivariate analysis. Correlation can be calculated with method ***corr()*** in pandas.




```
plt.figure(figsize=(15,7))
sns.heatmap(df.corr(), annot=True, linewidths=0.5,linecolor="black", fmt='.2f')
```

```
<AxesSubplot:>
```



9- Splitting data into Independent Features and Dependent Features:

For feature importance and feature scaling, we need to split data into independent and dependent features.

```
X = df_new.loc[:,df_new.columns != "RainTomorrow"]
y = df_new.loc[:,["RainTomorrow"]]
```

In the above code,

- *X* – Independent Features *or* Input features
- *y* – Dependent Features *or* target label

10- Feature Importance:

- Machine Learning Model performance depends on features that are used to train a model. **Feature importance** describes which features are relevant to build a model.
- **Feature Importance** refers to the techniques that assign a score to input/label features based on how useful they are at predicting a target variable. Feature importance helps in **Feature Selection.**
- We'll be using **ExtraTreesRegresso**r class for Feature Importance. This class implements a meta estimator that fits a number of randomized decision trees on various samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

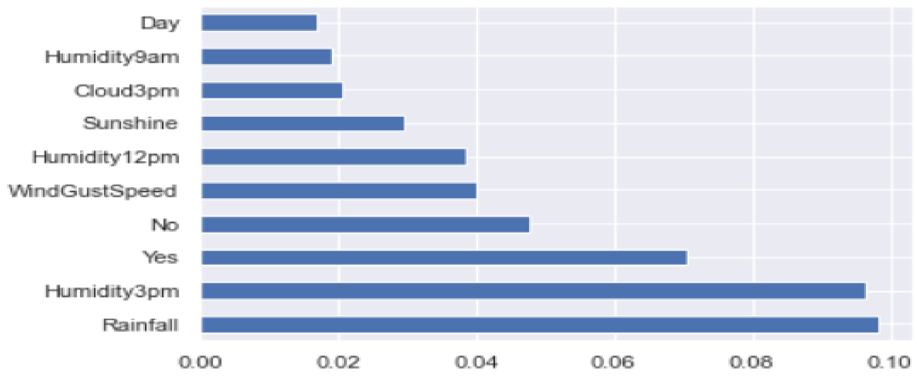
```
from sklearn.ensemble import ExtraTreesRegressor
etr_model = ExtraTreesRegressor()
etr_model.fit(X,y)
etr_model.feature_importances_
```

```
array([1.24198672e-02, 1.57425097e-02, 9.82319348e-02, 1.27137018e-02,
       2.96772872e-02, 3.99785725e-02, 1.44891451e-02, 1.50247643e-02,
       1.90685562e-02, 9.65275843e-02, 1.39432010e-02, 1.61530227e-02,
       1.24205420e-02, 2.05996849e-02, 1.17017901e-02, 1.20835731e-02,
       1.22862145e-02, 1.42356812e-02, 1.68093679e-02, 2.12615725e-03,
       3.06782756e-03, 3.39121678e-03, 5.71890719e-03, 3.80200204e-03,
       5.34663490e-03, 3.45880480e-03, 6.11148509e-03, 6.71377133e-03,
       5.32693144e-03, 4.25089065e-03, 4.57498923e-03, 5.96101696e-03,
       4.19140203e-03, 4.94555883e-03, 3.36582535e-03, 2.42746657e-03,
       3.10341362e-03, 2.23188440e-03, 8.17286567e-03, 3.32653771e-03,
       5.56929968e-03, 3.92437504e-03, 5.73372913e-03, 6.73053649e-03,
       4.56340118e-03, 3.95092460e-03, 5.98702409e-03, 5.18291172e-03,
       5.25672525e-03, 5.47392029e-03, 4.77879294e-03, 5.89237174e-03,
       8.18819868e-03, 3.97982608e-03, 6.08061075e-03, 5.10563583e-03,
       4.25761423e-03, 8.04281981e-03, 5.59177862e-03, 9.72075782e-03,
       6.75740237e-03, 4.97078687e-03, 3.74110652e-03, 5.44844194e-03,
       3.49197750e-03, 4.11739043e-03, 3.76998205e-03, 4.78193056e-02,
       7.07053257e-02, 2.28245252e-03, 1.09497256e-02, 3.30989284e-03,
       3.80771033e-03, 7.33272906e-04, 1.22252775e-02, 1.47392040e-03,
       2.78270252e-03, 3.82499220e-03, 5.08672888e-05, 8.80469641e-03,
       3.26919917e-03, 1.14603677e-02, 1.59927196e-02, 3.84931455e-02,
       1.42818581e-02, 1.52614420e-02, 1.04401948e-02])
```

Let's visualize feature importance values:

```
feature_imp = pd.Series(etr_model.feature_importances_,index=X.columns)
feature_imp.nlargest(10).plot(kind='barh')
```

```
<AxesSubplot:>
```



11- Splitting Data into training and testing set:

train_test_split () is a method of model selection class used to split data into training and testing sets.

```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, train_size=0.80, random_state=2, )
print(Xtrain.shape)
print(Xtest.shape)
print(ytrain.shape)
print(ytest.shape)
```

```
(6740, 87)
(1685, 87)
(6740, 1)
(1685, 1)
```

```
print(Xtrain.shape)
print(ytrain.shape)
```

```
(6740, 87)
(6740, 1)
```

12- Feature Scaling:

Feature Scaling is a technique used to scale, normalize, standardize data in range (0,1). When each column of a dataset has distinct values, then it helps to scale data of each column to a common level. **StandardScaler** is a class used to implement feature scaling.

```
scaler = StandardScaler()
#fitting standardization on train data only
scaler.fit(Xtrain)
XtrainSTD = scaler.transform(Xtrain)
XtestSTD = scaler.transform(Xtest)
```

```
XtrainSTD.shape
```

```
(6740, 87)
```

13- Model Building

Different algorithms can be used for making the predictive model. I'll be using simple logistic regression for demonstrations. A similar approach can be used for applying more sophisticated algorithms like random forest, decision trees, XGBoost, etc.

Random Forest Classifier:

A simple version of random forest classifier without changing the parameter settings is applied to the training test and later on evaluated by using it on the test set.

Your content goes here. Edit or remove this text inline or in the module Content settings. You can also style every aspect of this content in the module Design settings and even apply custom CSS to this text in the module Advanced settings.

```
rf = RandomForestClassifier(random_state=0)
parameters = {'bootstrap': [True, False],
              'min_samples_split':[2, 3, 4],
              'criterion':['entropy', 'gini'],
              'n_estimators':[100, 200]
             }
grid_search1 = GridSearchCV(estimator=rf, param_grid=parameters, refit='roc_auc', scoring=['accuracy', 'roc_auc'], cv=10, n_jobs=
grid_search1 = grid_search1.fit(XtrainSTD, ytrain.values.ravel())
```

```
print("Best Parameters : ",grid_search1.best_params_)
print("Best AUC-ROC : ", grid_search1.best_score_)
```

```
Best Parameters :  {'bootstrap': True, 'criterion': 'entropy', 'min_samples_split': 2, 'n_estimators': 100}
Best AUC-ROC :  nan
```

14- Model Evaluation

Different evaluation metrics can be used based on the problem and industry. In this case, the accuracy score has been used. The accuracy Score of training and testing data is comparable and almost equal. So, there is no question of underfitting and overfitting. And the model is generalizing well for new unseen data.

```
rf = RandomForestClassifier(bootstrap= False, criterion= 'entropy', min_samples_split= 4, n_estimators= 200, random_state=0)
rf.fit(XtrainSTD, ytrain.values.ravel())
ypred = rf.predict(XtestSTD)
accuracy = accuracy_score(ypred, ytest)
print(accuracy)
```

```
0.8896142433234422
```

This algorithm has a predictive accuracy of **88.96.**

15- Results and Conclusion:

- The random forest classifier model accuracy score is 0.88. The model does a very good job of predicting.
- The model shows no sign of Underfitting or Overfitting. This means the model generalizing well for unseen data.
- The mean accuracy score of cross-validation is almost the same as the original model accuracy score**.** So, the accuracy of the model may not be improved using Cross-validation.

- **About the Author:**
- Hello, I'm **Abhinay R. Gudadhe,** pursuing a Master of Technology in Power Electronics & Drives Engineering from **G.H Raisoni College of Engineering**, Nagpur (Maharashtra).