Weather Prediction

Will it rain Tomorrow?

Muhammed Berk Önder

İstanbul Data Science Academy

January 21, 2022

Purpose

Will it rain tomorrow?

- **Problem:** Can we tell from today that it will rain tomorrow?
- **Whom does it concern:** Anyone who doesn't want to get wet tomorrow.

Methodology

- Pull sql table from database with *sqlalchemy*.
- Analyse and Clean the data.
- Used *RandomForestClassifier* as a model.
- Deploy with *Flask*.

```
df.info()
```

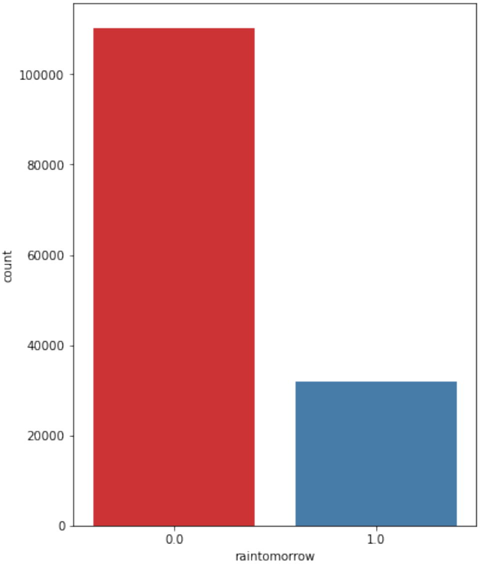
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   date           145460 non-null  datetime64[ns]
 1   location       145460 non-null  object
 2   mintemp        143975 non-null  float64
 3   maxtemp        144199 non-null  float64
 4   rainfall       142199 non-null  float64
 5   evaporation    82670 non-null   float64
 6   sunshine       75625 non-null   float64
 7   windgustdir    135134 non-null  object
 8   windgustspeed  135197 non-null  float64
 9   winddir9am     134894 non-null  object
 10  winddir3pm     141232 non-null  object
 11  windspeed9am   143693 non-null  float64
 12  windspeed3pm   142398 non-null  float64
 13  humidity9am    142806 non-null  float64
 14  humidity3pm    140953 non-null  float64
 15  pressure9am    130395 non-null  float64
 16  pressure3pm    130432 non-null  float64
 17  cloud9am       89572 non-null   float64
 18  cloud3pm       86102 non-null   float64
 19  temp9am        143693 non-null  float64
 20  temp3pm        141851 non-null  float64
 21  raintoday      142199 non-null  object
 22  raintomorrow   142193 non-null  object
dtypes: datetime64[ns](1), float64(16), object(6)
memory usage: 25.5+ MB
```

```
df.dtypes
```

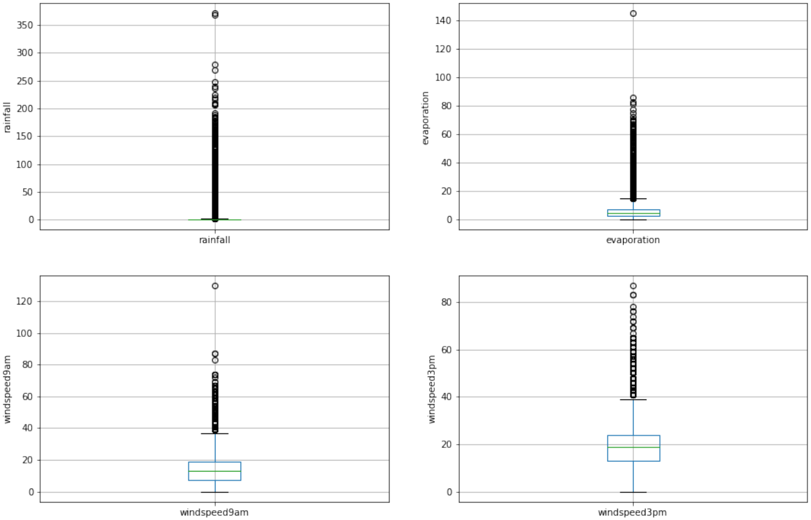
```
date             datetime64[ns]
location                 object
mintemp                 float64
maxtemp                 float64
rainfall                float64
evaporation             float64
sunshine                float64
windgustdir              object
windgustspeed           float64
winddir9am               object
winddir3pm               object
windspeed9am            float64
windspeed3pm            float64
humidity9am             float64
humidity3pm             float64
pressure9am             float64
pressure3pm             float64
cloud9am                float64
cloud3pm                float64
temp9am                 float64
temp3pm                 float64
raintoday                object
raintomorrow             object
dtype: object
```

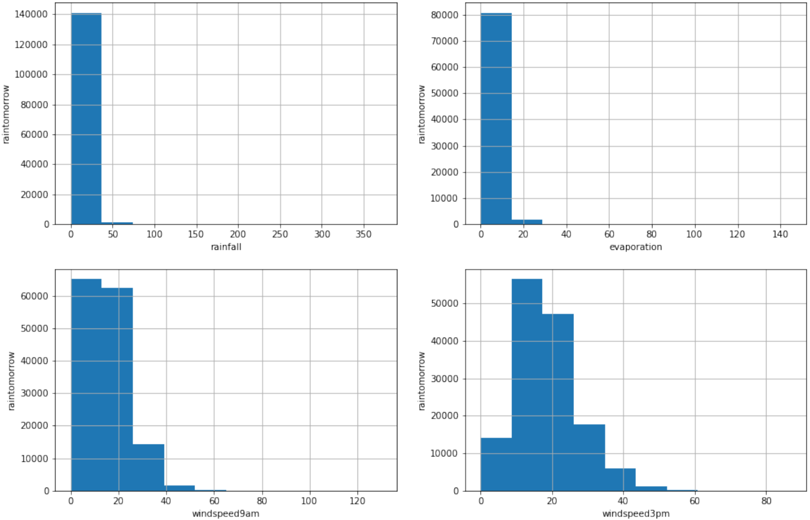
```
df.isnull().sum()
```

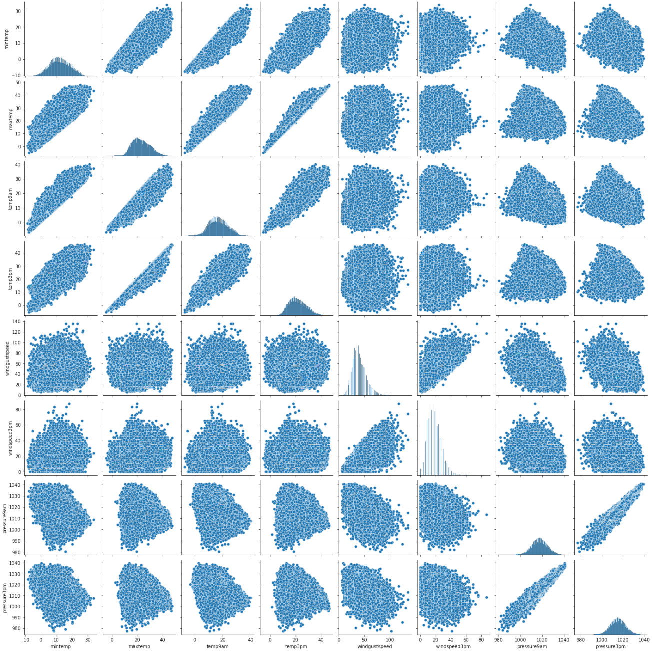
```
date                 0
location             0
mintemp           1485
maxtemp           1261
rainfall          3261
evaporation      62790
sunshine         69835
windgustdir      10326
windgustspeed    10263
winddir9am       10566
winddir3pm        4228
windspeed9am      1767
windspeed3pm      3062
humidity9am       2654
humidity3pm       4507
pressure9am      15065
pressure3pm      15028
cloud9am         55888
cloud3pm         59358
temp9am           1767
temp3pm           3609
raintoday         3261
raintomorrow      3267
dtype: int64
```



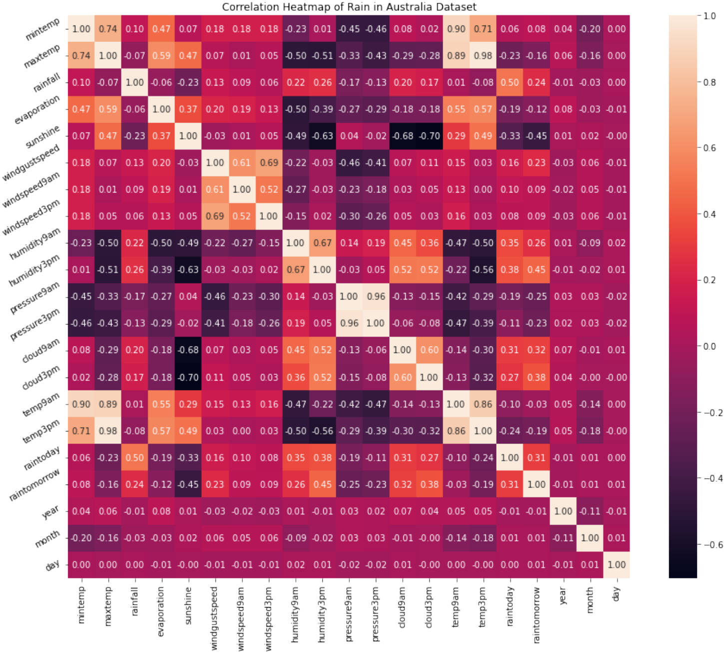
100000
80000
60000
40000
20000
0
count
0.0
1.0
raintomorrow


rainfall
0
50
100
150
200
250
300
350
rainfall
evaporation
0
20
40
60
80
100
120
140
evaporation
windspeed9am
0
20
40
60
80
100
120
windspeed9am
windspeed3pm
0
20
40
60
80
windspeed3pm


raintomorrow
rainfall
raintomorrow
evaporation
raintomorrow
windspeed9am
raintomorrow
windspeed3pm



mintemp
maxtemp
temp9am
temp3pm
windgustspeed
windspeed3pm
pressure9am
pressure3pm


Correlation Heatmap of Rain in Australia Dataset
mintemp
maxtemp
rainfall
evaporation
sunshine
windgustspeed
windspeed9am
windspeed3pm
humidity9am
humidity3pm
pressure9am
pressure3pm
cloud9am
cloud3pm
temp9am
temp3pm
raintoday
raintomorrow
year
month
day
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

From the above correlation heat map, we can conclude that:

- MinTemp and MaxTemp variables are highly positively correlated (correlation coefficient = 0.74).
- MinTemp and Temp3pm variables are also highly positively correlated (correlation coefficient = 0.71).
- MinTemp and Temp9am variables are strongly positively correlated (correlation coefficient = 0.90).
- MaxTemp and Temp9am variables are strongly positively correlated (correlation coefficient = 0.89).
- MaxTemp and Temp3pm variables are also strongly positively correlated (correlation coefficient = 0.98).
- WindGustSpeed and WindSpeed3pm variables are highly positively correlated (correlation coefficient = 0.69).
- Pressure9am and Pressure3pm variables are strongly positively correlated (correlation coefficient = 0.96).
- Temp9am and Temp3pm variables are strongly positively correlated (correlation coefficient = 0.86).

```python
def max_value(df4, variable, top):
    return np.where(df4[variable]>top, top, df4[variable])

for df4 in [X_train, X_test]:
    df4['rainfall'] = max_value(df4, 'rainfall', 3.2)
    df4['evaporation'] = max_value(df4, 'evaporation', 21.8)
    df4['windspeed9am'] = max_value(df4, 'windspeed9am', 55)
    df4['windspeed3pm'] = max_value(df4, 'windspeed3pm', 57)
```

```python
print('Rainfall:', X_train.rainfall.max(), X_test.rainfall.max())
print('Evaporation:', X_train.evaporation.max(), X_test.evaporation.max())
print('WindSpeed9am:', X_train.windspeed9am.max(), X_test.windspeed9am.max())
print('WindSpeed3pm:', X_train.windspeed3pm.max(), X_test.windspeed3pm.max())

print(X_train[numerical].describe())
```

```
Rainfall: 3.2 3.2
Evaporation: 21.8 21.8
WindSpeed9am: 55.0 55.0
WindSpeed3pm: 57.0 57.0
              mintemp         maxtemp        rainfall     evaporation  \
count  116368.000000  116368.000000  116368.000000  116368.000000
mean       12.190148      23.203007       0.670632       5.093247
std         6.366878       7.085492       1.181365       2.800193
min        -8.500000      -4.800000       0.000000       0.000000
25%         7.700000      18.000000       0.000000       4.000000
50%        12.000000      22.600000       0.000000       4.700000
75%        16.800000      28.200000       0.600000       5.200000
max        31.900000      48.100000       3.200000      21.800000
```

```python
# impute missing values in X_train and X_test with respective column median in X_train
for df1 in [X_train, X_test]:
    for col in numerical:
        col_median=X_train[col].median()
        df1[col].fillna(col_median, inplace=True)
```

```python
# check again missing values in numerical variables in X_train
X_train[numerical].isnull().sum()
```

```
mintemp          0
maxtemp          0
rainfall         0
evaporation      0
sunshine         0
windgustspeed    0
windspeed9am     0
windspeed3pm     0
humidity9am      0
humidity3pm      0
pressure9am      0
pressure3pm      0
cloud9am         0
cloud3pm         0
temp9am          0
temp3pm          0
raintoday        0
year             0
month            0
day              0
dtype: int64
```

```python
# impute missing categorical variables with most frequent value
for df2 in [X_train, X_test]:
    df2['windgustdir'].fillna(X_train['windgustdir'].mode()[0], inplace=True)
    df2['winddir9am'].fillna(X_train['winddir9am'].mode()[0], inplace=True)
    df2['winddir3pm'].fillna(X_train['winddir3pm'].mode()[0], inplace=True)
```

```python
# check missing values in categorical variables in X_train
X_train[categorical].isnull().sum()
```

```
location       0
windgustdir    0
winddir9am     0
winddir3pm     0
dtype: int64
```

```
X_test = pd.concat([X_test[numerical],
                    pd.get_dummies(X_test.location),
                    pd.get_dummies(X_test.windgustdir),
                    pd.get_dummies(X_test.winddir9am),
                    pd.get_dummies(X_test.winddir3pm)], axis=1)
```

```python
results = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent',
              'Decision Tree'],
    'Score': [acc_linear_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_decision_tree]})
result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

	Model
Score	
100.00	KNN
100.00	Random Forest
100.00	Decision Tree
84.66	Logistic Regression
84.59	Support Vector Machines
81.05	Stochastic Gradient Decent
73.89	Perceptron
64.37	Naive Bayes

```
# visualize confusion matrix with seaborn heatmap
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:0'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')

from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_test))

TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91     22730
         1.0       0.73      0.49      0.59      6362

    accuracy                           0.85     29092
   macro avg       0.80      0.72      0.75     29092
weighted avg       0.84      0.85      0.84     29092
```