Supervised Machine Learning: Classification

Final Peer Assignment

Objective of the analysis

A goal of this project is focused on Classification models to rain in Canberra, Australia. Simple logistic regression, KNN and decision tree models were created. Also, models (regression, KNN and decision tree) with GridSearchCV were built to find best parameters for classifiers to analyze which model suites best to a dataset. This project focused on prediction only on one Australian city, but it is also possible to create a project with target variable "Location" to predict where there will be rainy weather.

Dataset description

The dataset "weatherAUS" was downloaded from a Kaggle.com (https://www.kaggle.com/jsphyg/weather-dataset-rattle-package). This dataset contains about 10 years of daily weather observations from many locations across Australia. This dataset originally consists of 145460 rows and 23 columns with following features:

'Date' – date of an observation;

'Location' – location of an observation;

'MinTemp' - Minimum temperature in the 24 hours to 9am (degrees Celsius);

'MaxTemp' - Maximum temperature in the 24 hours from 9am (degrees Celsius);

'Rainfall' - Precipitation (rainfall) in the 24 hours to 9am (millimeters);

'Evaporation' - "Class A" pan evaporation in the 24 hours to 9am (millimeters);

'Sunshine' – Bright sunshine in the 24 hours to midnight (hours);

'WindGustDir' - Direction of strongest gust in the 24 hours to midnight (16 compass points);

'WindGustSpeed' - Speed of strongest wind gust in the 24 hours to midnight (kilometers per hour);

'WindDir9am' - Wind direction averaged over 10 minutes prior to 9 am (compass points);

'WindDir3pm' - Wind direction averaged over 10 minutes prior to 3 pm (compass points);

'WindSpeed9am' - Wind speed averaged over 10 minutes prior to 9 am (kilometers per hour);

'WindSpeed3pm' - Wind speed averaged over 10 minutes prior to 3pm (kilometers per hour);

'Humidity9am' - Humidity at 9 am (percent);

'Humidity3pm' - Humidity at 3 pm (percent);

'Pressure9am' - Atmospheric pressure reduced to mean sea level at 9 am (hpa)

'Pressure3pm' - Atmospheric pressure reduced to mean sea level at 3pm (hpa)

'Cloud9am' - Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.

'Cloud3pm' - Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.

'Temp9am' - Temperature at 9am (degrees Celsius);

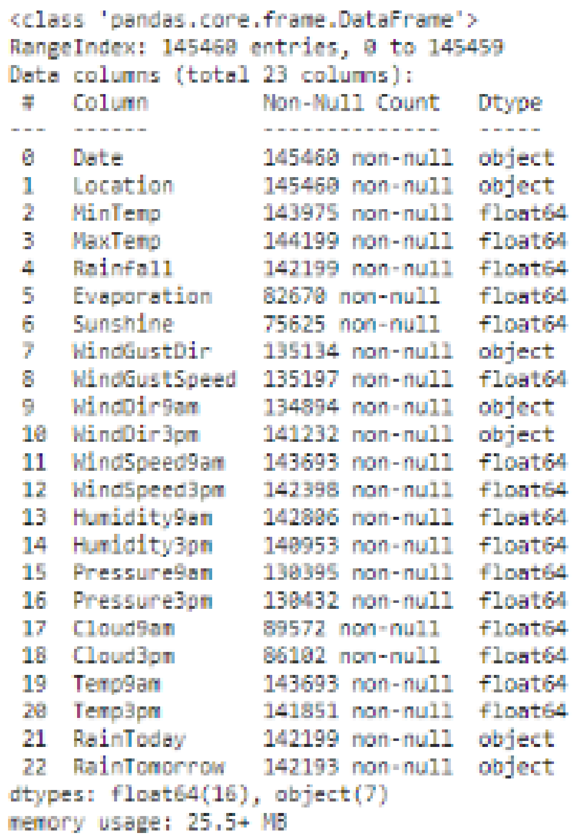
'Temp3pm' - Temperature at 3 pm (degrees Celsius);

'RainToday' - Boolean: "yes" or "no";

'RainTomorrow' - Boolean: “yes” or “no”. Target variable.

Data exploration analysis

First, all necessary libraries were imported (e.g. LogisticRegression, DecisionTreeClassifier and etc.) First step in EDA is to get familiar with the data set, for this purpose methods .head(), .shape and .columns were used. As it was mentioned in previous section, the dataset contains 145460 rows with 23 features. Method .info() showed type (float, object) of each column and it’s missed values. As it shown on the Figure 1, some columns, such as “Sunshine” or “Evaporation”, have a lot of missed values.



```
<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  object
 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: float64(16), object(7)
memory usage: 25.5+ MB
```

Figure 1. Results of .info() method

But as far as this project is focused on rain prediction only in one location, therefore command **pd. value_counts(df["Location"])**

was used to find out how much unique values and observations per value contains the column “Location”. Top 3 cities a list are: Canberra has 3436 counts; Sydney has 3344 and Melbourne 3193. Therefore, further work was done only with Canberra.

For this purpose, new DataFrame with the use **df_canberra = df[df["Location"] == "Canberra"]** of was created. To see how much missing values are in each column a following loop was created:

```
missing_data = df_canberra.isnull()
for column in missing_data.columns.values.tolist():
   print(column)
   print (missing_data[column].value_counts())
   print("")
```

It shows, that following columns contain too much of missing data: "Evaporation", "Sunshine", "Date", "Cloud9am", "Cloud3pm". Therefore, these features were excluded. Columns "WindGustDir","WindDir9am", "WindDir3pm" also contain a lot of missing values (e.g. 532, 320 and etc. missing values out of 3402 total count). In this case, the problem is that it is not possible to fill with predicted, mean or median values (because features are objects) or drop rows, because than a lot of information would be lost. The only solution here is completely drop these columns. Features "RainToday","RainTomorrow" have only 18 missing values each, the last feature is our target, which can not be dropped or contain NaN values. Here rows with missing values were dropped.

Another situation with float features, there were not so much missing values (e.g. 17, 12 or 7). A solution here was to replace NaN with mean values using following loop:

```
for column in df_canberra.select_dtypes(exclude=['object']):
    print(column)
    print (df_canberra[column].mean())
    df_canberra[column] = df_canberra[column].replace(np.nan,
    df_canberra[column].mean())
    print("Done")
df_canberra.head()
print(df_canberra.shape)
```

After cleaning Canberra dataset contains 3402 rows and following features: 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'RainTomorrow'. Figure 2 represents correlation matrix between different features.

Features 'RainToday', 'RainTomorrow' were encoded with the use of LabelEncoder. Dataframe was splitted into train and test splits. These splits were used for all models.

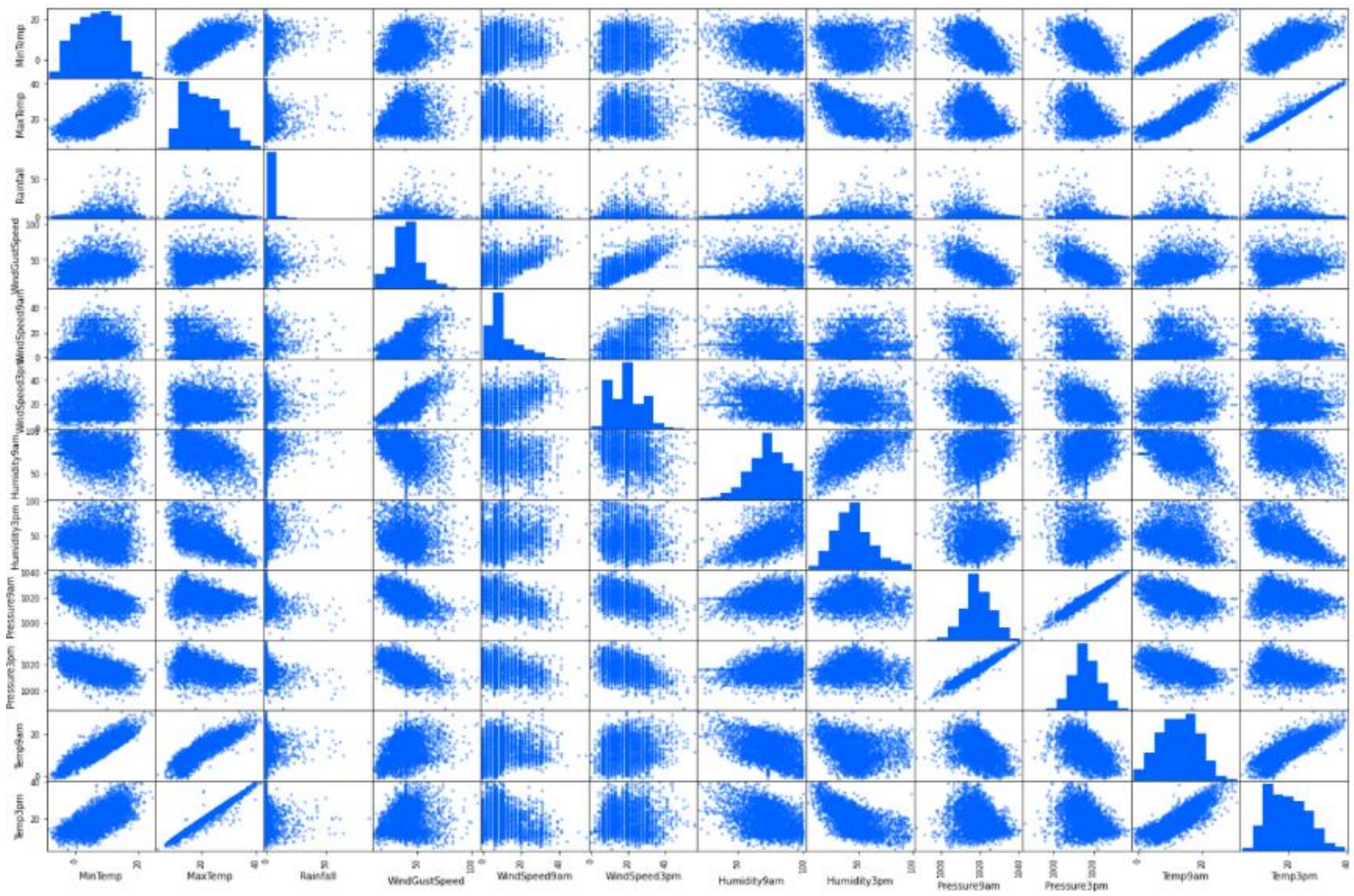
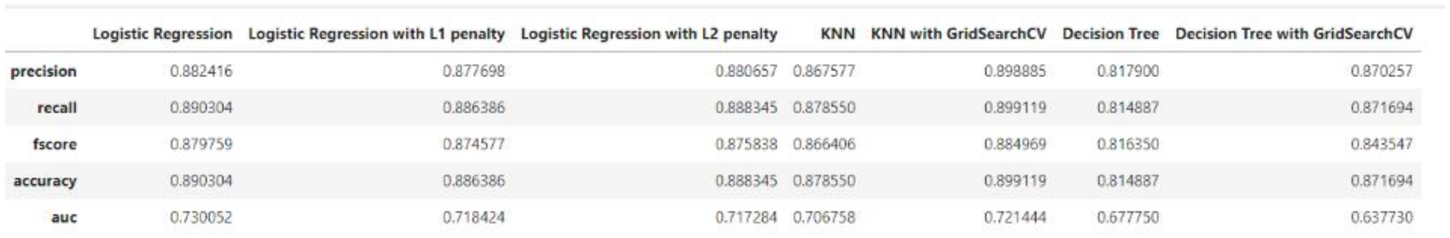


Figure 2. Correlation matrix of different features in Canberra dataset

<u>Classifications</u>

In this project the following models were build: Logistic Regression as a base model, K Nearest Neighbor and Decision Tree. For each classification method a "general" model and model with parameter tuning were created to see which model suites better. Following scores were calculated: precision, recall, fscore, accuracy, AUC as well as confusion matrixes.



	Logistic Regression	Logistic Regression with L1 penalty	Logistic Regression with L2 penalty	KNN	KNN with GridSearchCV	Decision Tree	Decision Tree with GridSearchCV
precision	0.882416	0.877698	0.880657	0.867577	0.898885	0.817900	0.870257
recall	0.890304	0.886386	0.888345	0.878550	0.899119	0.814887	0.871694
fscore	0.879759	0.874577	0.875838	0.866406	0.884969	0.816350	0.843547
accuracy	0.890304	0.886386	0.888345	0.878550	0.899119	0.814887	0.871694
auc	0.730052	0.718424	0.717284	0.706758	0.721444	0.677750	0.637730

Figure 3. Score results for different classification models

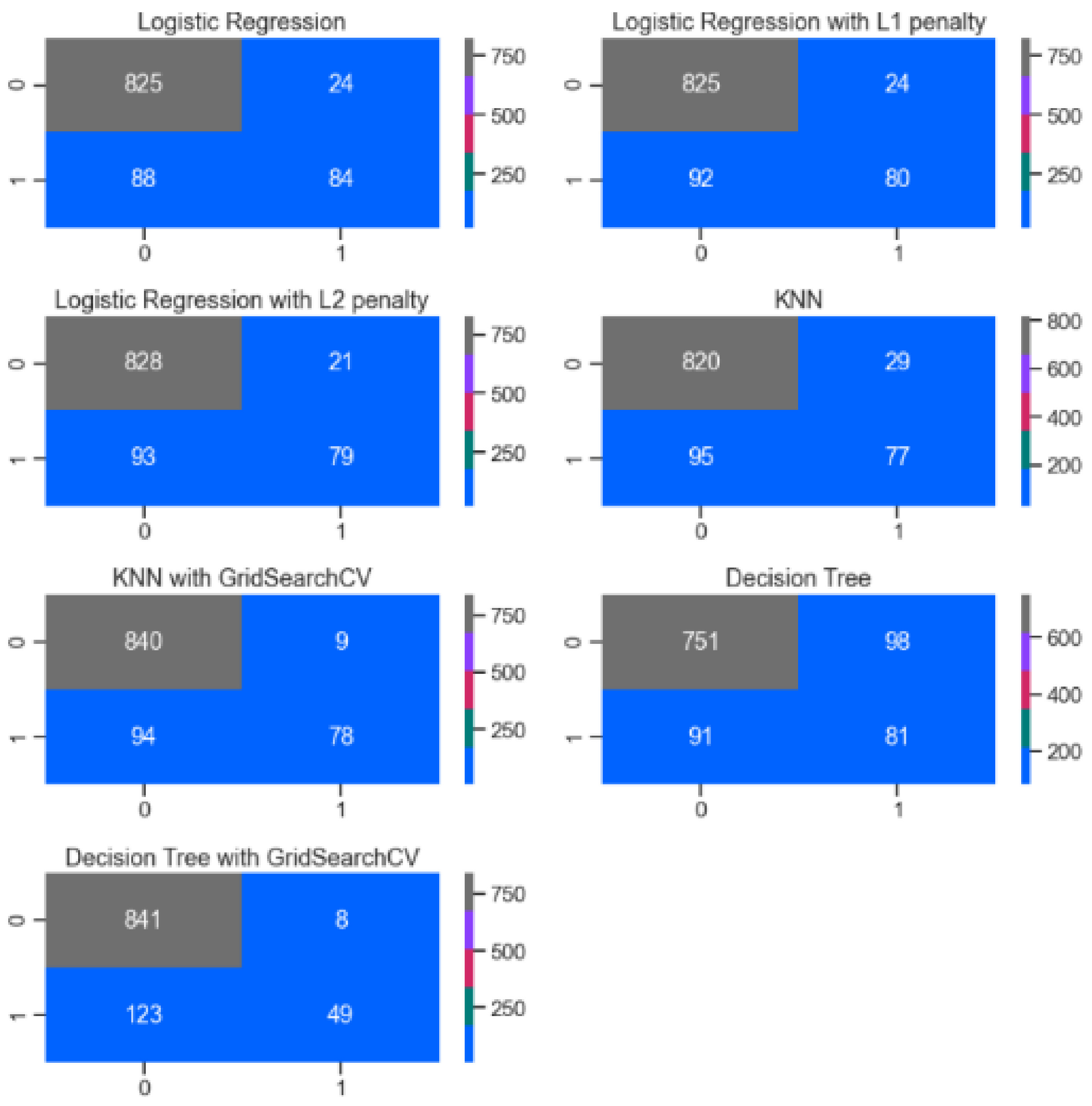


Figure 4. Confusion matrixes for different classification models

As it is shown in Figure 3, all models gave the same scores with slight differences. But the best model is KNN with n- neighbors equals to 10. Simple Logistic Regression is slightly better than Logistic Regression with L2 penalty. Decision Tree models gave the "worthiest" results. Simple decision Tree model has 599 nods and tree depth is 20. With GridSearchCV number of nods and depth were reduced to 15 and 3, respectively.

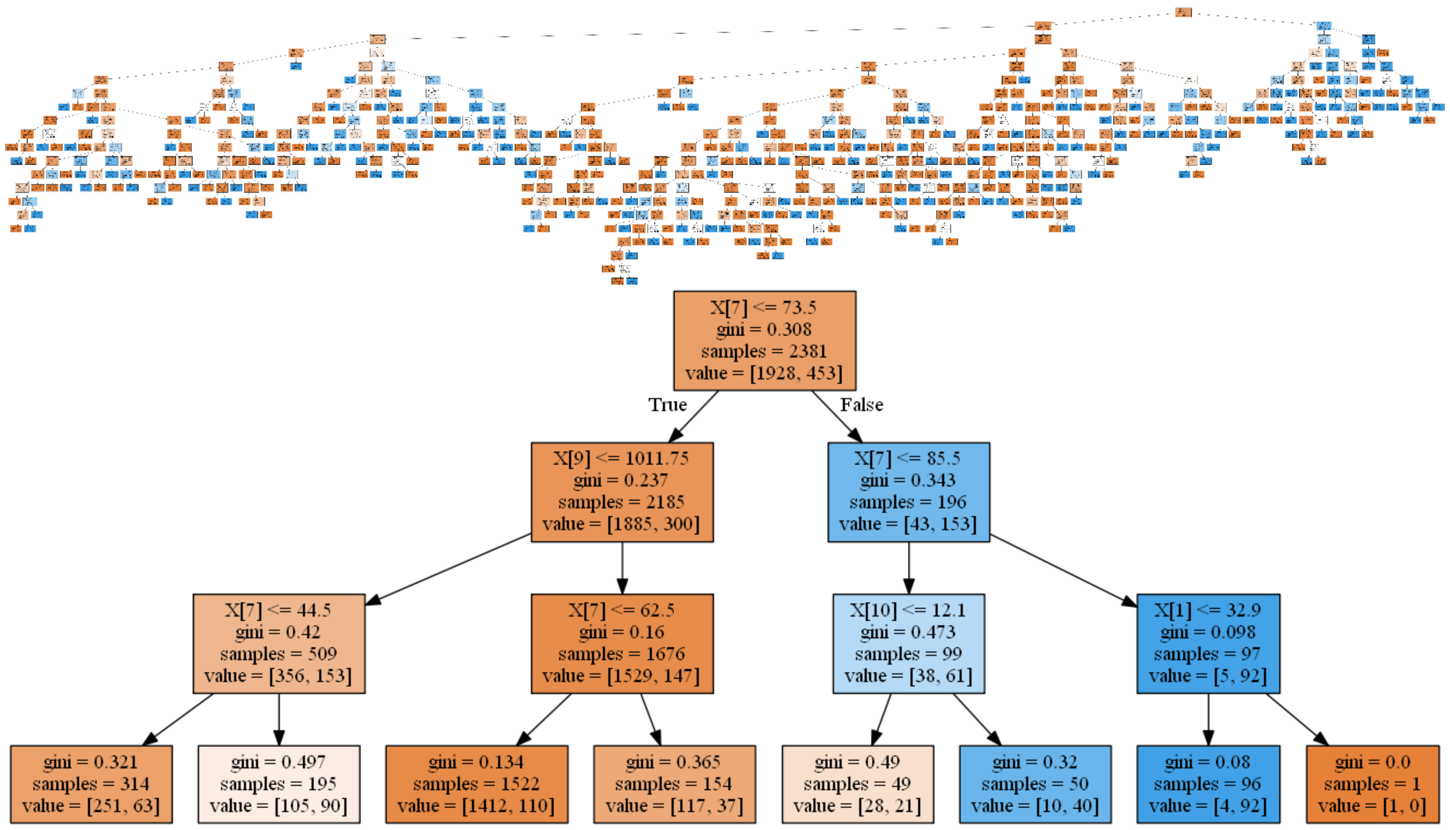


Figure 5. Decision Tree models without (upper) and with GridSearchCV(down)

Conclusion

In this project the Canberra data set was used. It was cleaned and prepared for further work for different types of classification models. Logistic Regression, K Nearest Neighbor and Decision Tree model were created. The best results were obtained for GridSearchCV KNN model with accuracy 0.899. The second-best result has Simple Logistic regression with 0.890 accuracy. Decision Tree models have 0.87 accuracy, which is the lowest result.

To get better results it is necessary to additional information which could help to get higher results. In this case, such features as "Evaporation", "Sunshine", "Cloud9am", "Cloud3pm", "WindGustDir","WindDir9am", "WindDir3pm" which were excluded, which were excluded because of lack of observation, could give us more full information and better predictions. Together with getting more correlated feature, I would also try to use more advanced techniques such as ensemble based methods.