AUTOMATION MACHINE LEARNNG

Final Project Report

Arranged by :

1301184423 – Diya Namira Purba

1301183625 – Inacio Campos

Bachelor of Informatics

School of Computing
Telkom University
Bandung

Automation Machine Learning

1. **Formulation of Problem**
    - Articulate Your Problem Clearly
      Daily weather observations from multiple locations around Australia. This experiment will be solved using TPOT automation tools for machine learning
    - Identify Your Data Sources
      This dataset contains 10 years of weather observation from Australia.
    - Identify Your Learning Problem
      Automation machine learning using TPOT automation tools
    - Think about Potential Bias and Ethics
      The dataset has a target variable that is RainTomorrow. All features will be used in this experiment except categorical features such as Date, Location, WindgustDir, WinDir9am, and WinDir3pm
2. **Exploration and Preparation of Data**
    - Missing Values
      Check missing values in dataset and impute the missing values. Filling in missing values to prevent changes in the amount of data in the datasetthat could affect the process.

▾ Check Missing Values

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

```
Date              0
Location          1
MinTemp          48
MaxTemp          41
Rainfall        115
Evaporation    6085
Sunshine       6858
WindGustDir     127
WindGustSpeed   127
WindDir9am      968
WindDir3pm      135
WindSpeed9am     63
WindSpeed3pm     62
Humidity9am      67
Humidity3pm      66
Pressure9am     181
Pressure3pm     187
Cloud9am       4796
Cloud3pm       4651
Temp9am          54
Temp3pm          55
RainToday       115
RainTomorrow    115
dtype: int64
```

```
[ ] df["MinTemp"]= df["MinTemp"].fillna(df["MinTemp"].mean())
    df["MaxTemp"]= df["MaxTemp"].fillna(df["MaxTemp"].mean())
    df["Evaporation"]= df["Evaporation"].fillna(df["Evaporation"].mean())
    df["Sunshine"]= df["Sunshine"].fillna(df["Sunshine"].mean())
    df["WindGustSpeed"]= df["WindGustSpeed"].fillna(df["WindGustSpeed"].mean())
    df["Rainfall"]= df["Rainfall"].fillna(df["Rainfall"].mean())
    df["WindSpeed9am"]= df["WindSpeed9am"].fillna(df["WindSpeed9am"].mean())
    df["WindSpeed3pm"]= df["WindSpeed3pm"].fillna(df["WindSpeed3pm"].mean())
    df["Humidity9am"]= df["Humidity9am"].fillna(df["Humidity9am"].mean())
    df["Humidity3pm"]= df["Humidity3pm"].fillna(df["Humidity3pm"].mean())
    df["Pressure9am"]= df["Pressure9am"].fillna(df["Pressure9am"].mean())
    df["Pressure3pm"]= df["Pressure3pm"].fillna(df["Pressure3pm"].mean())
    df["Cloud9am"]= df["Cloud9am"].fillna(df["Cloud9am"].mean())
    df["Cloud3pm"]= df["Cloud3pm"].fillna(df["Cloud3pm"].mean())
    df["Temp9am"]= df["Temp9am"].fillna(df["Temp9am"].mean())
    df["Temp3pm"]= df["Temp3pm"].fillna(df["Temp3pm"].mean())



    df['WindDir9am'] = df['WindDir9am'].fillna(df['WindDir9am'].mode()[0])
    df['WindGustDir'] = df['WindGustDir'].fillna(df['WindGustDir'].mode()[0])
    df['WindDir3pm'] = df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0])
```

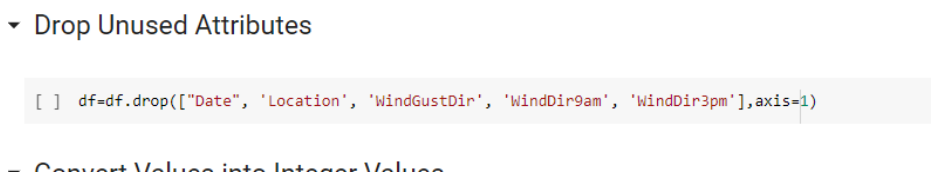
- **Encoding**

  For RainToday and RainTommorrow attributes the values will be convert into

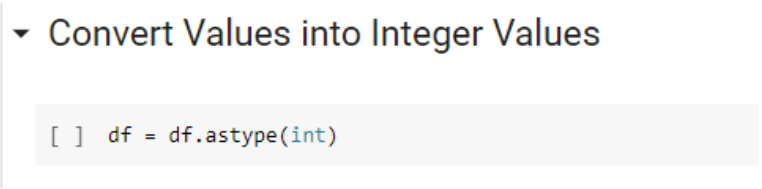
  0 and 1. The missing values will fill with 0 to make the values same with Convert values

```
[ ]
    df["RainToday"] = df["RainToday"].str.replace("No", "0")
    df["RainToday"] = df["RainToday"].str.replace("Yes", "1")

    df["RainTomorrow"] = df["RainTomorrow"].str.replace("No", "0")
    df["RainTomorrow"] = df["RainTomorrow"].str.replace("Yes", "1")

    df["RainTomorrow"]= df["RainTomorrow"].fillna(0)
    df["RainToday"]= df["RainToday"].fillna(0)
```

- Drop Unused Attributes
  Attributes that are not used will be dropped so that the process avoids errors. The dropped attributes such as date, location, windgustdir, windir9am, and windir3pm.



- Convert Dataset into Integer Values
  The dataset has a float value but will be converted into an integer value because the existing process can only process integer values not floats (error).



- Train and Test Data Split
  The X data contains all attributes except RainTomorrow attribute and y data only contains RainTomorrow. the two data will be split into x_train, x_test, y_train, and y_test.

```
[ ] from sklearn.model_selection import train_test_split

    X = df.drop('RainTomorrow', axis=1)
    y = df['RainTomorrow']

    x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

3. Modelling

In this automation machine learning will be used TPOT and weatherAUS.CSV dataset.

4. Evaluation

Evaluation of the model will be used the TPOT score from the test data and TPOT classification score from train data.

5. Experiment

Import library and read the datasets.

▾ Import Library

```
[ ] import pandas as pd
    from google.colab import files

    #uploaded = files.upload()
```

▾ Read Data

```
[ ] df = pd.read_csv('weatherAUS.csv')
```

```
[ ] df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE

Check missing values and Impute the missing values with mean data if it's numerical values and mode if it's categorical value.

▾ Check Missing Values

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

```
Date              0
Location          1
MinTemp          48
MaxTemp          41
Rainfall        115
Evaporation    6085
Sunshine       6858
WindGustDir     127
WindGustSpeed   127
WindDir9am      968
WindDir3pm      135
WindSpeed9am     63
WindSpeed3pm     62
Humidity9am      67
Humidity3pm      66
Pressure9am     181
Pressure3pm     187
Cloud9am       4796
Cloud3pm       4651
Temp9am          54
Temp3pm          55
RainToday       115
RainTomorrow    115
dtype: int64
```

```
[ ] df["MinTemp"]= df["MinTemp"].fillna(df["MinTemp"].mean())
    df["MaxTemp"]= df["MaxTemp"].fillna(df["MaxTemp"].mean())
    df["Evaporation"]= df["Evaporation"].fillna(df["Evaporation"].mean())
    df["Sunshine"]= df["Sunshine"].fillna(df["Sunshine"].mean())
    df["WindGustSpeed"]= df["WindGustSpeed"].fillna(df["WindGustSpeed"].mean())
    df["Rainfall"]= df["Rainfall"].fillna(df["Rainfall"].mean())
    df["WindSpeed9am"]= df["WindSpeed9am"].fillna(df["WindSpeed9am"].mean())
    df["WindSpeed3pm"]= df["WindSpeed3pm"].fillna(df["WindSpeed3pm"].mean())
    df["Humidity9am"]= df["Humidity9am"].fillna(df["Humidity9am"].mean())
    df["Humidity3pm"]= df["Humidity3pm"].fillna(df["Humidity3pm"].mean())
    df["Pressure9am"]= df["Pressure9am"].fillna(df["Pressure9am"].mean())
    df["Pressure3pm"]= df["Pressure3pm"].fillna(df["Pressure3pm"].mean())
    df["Cloud9am"]= df["Cloud9am"].fillna(df["Cloud9am"].mean())
    df["Cloud3pm"]= df["Cloud3pm"].fillna(df["Cloud3pm"].mean())
    df["Temp9am"]= df["Temp9am"].fillna(df["Temp9am"].mean())
    df["Temp3pm"]= df["Temp3pm"].fillna(df["Temp3pm"].mean())



    df['WindDir9am'] = df['WindDir9am'].fillna(df['WindDir9am'].mode()[0])
    df['WindGustDir'] = df['WindGustDir'].fillna(df['WindGustDir'].mode()[0])
    df['WindDir3pm'] = df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0])
```

Drop unused attributes and convert the dataset values into integer values.

Drop Unused Attributes

```
[ ]  df=df.drop(["Date", 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm'],axis=1)
```

Convert Values into Integer Values

```
[ ]  df = df.astype(int)
```

Create x and y data. Split the x and y data into x_train, x_test, y_train, and y_test.

```
[ ]  from sklearn.model_selection import train_test_split

     X = df.drop('RainTomorrow', axis=1)
     y = df['RainTomorrow']

     x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

This experiment used TPOT classifier for automation tools and train data will be used in this process. Generation score is 3, population score is 20 and verbosity is 2

```
tpot.fit(x_train, y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31: FutureWarning:
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarn:
  warnings.warn(message, FutureWarning)
Optimization Progress: 100%  80/80 [08:21<00:00, 6.51s/pipeline]

Generation 1 - Current best internal CV score: 0.8758010829579981

Generation 2 - Current best internal CV score: 0.8764757615215488

Generation 3 - Current best internal CV score: 0.8773180779985911

Best pipeline: GradientBoostingClassifier(Normalizer(input_matrix, norm=l2), learni
TPOTClassifier(config_dict=None, crossover_rate=0.1, cv=5,
               disable_update_check=False, early_stop=None, generations=3,
               log_file=None, max_eval_time_mins=5, max_time_mins=None,
               memory=None, mutation_rate=0.9, n_jobs=1, offspring_size=None,
               periodic_checkpoint_folder=None, population_size=20,
               random_state=None, scoring=None, subsample=1.0, template=None,
               use_dask=False, verbosity=2, warm_start=False)
```

For the TPOT score will be used test data and retrieve the TPOT result.

```
[ ]  print(tpot.score(x_test, y_test))

     0.8913630229419703

[ ]  tpot.export('result.py')
```

6. Conclusions

Based on the experiment evaluation it shows that the TPOT classifier produce three generations such as:

Generation 1 CV score is 0.87580

Generation 2 CV score is 0.8764

Generation 3 CV score is 0.8773

In the above, 3 generations were computed, each giving the training efficiency of the fitting model on the training set. As the evidence The best pipe line is gradient boosting classifier. It's also produced a high TPOT score is 0.8913 using test data.

7. Presentation Video

https://youtu.be/r8i-8xGxeP0

8. Source Code

https://colab.research.google.com/drive/1fKlUG657emKYD9IKHpXkJgUTC-5H6w-C?usp=sharing