Project 3: Weather

Mengyu Jackson

Overview

Business Problem

Data Understanding

In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pprint
import time

from sklearn.experimental import enable_iterative_imputer

from sklearn.pipeline import Pipeline
from sklearn.svm import SVC, NuSVC, LinearSVC
from sklearn.preprocessing import OneHotEncoder, FunctionTransformer, M
inMaxScaler, MaxAbsScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier,
ExtraTreesClassifier, AdaBoostClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier, Per
ceptron, PassiveAggressiveClassifier
from sklearn.neighbors import KNeighborsClassifier, RadiusNeighborsClas
sifier, NearestCentroid
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.impute import KNNImputer, SimpleImputer, IterativeImputer
from sklearn.pipeline import FeatureUnion
from sklearn.impute import MissingIndicator
```

Data Preparation

In [2]:
```
weatherAUS = pd.read_csv('./data/weatherAUS.csv')
df = weatherAUS.copy()
```

In [ ]:

In [ ]:

In [ ]:

In [3]:
```
weatherAUS.isna().sum()
```

Out[3]:
```
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
```

In [4]:
```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df.pop('Date')
df = df.dropna(subset=['RainToday', 'RainTomorrow'])
df['RainToday'] = df['RainToday'].replace('No', 0).replace('Yes', 1).as
type(float)
df['RainTomorrow'] = df['RainTomorrow'].replace('No', 0).replace('Yes',
1).astype(float)
```

In [5]:
```
df['WindGustDir'] = df['WindGustDir'].fillna("NaN")
df['WindDir9am'] = df['WindDir9am'].fillna("NaN")
df['WindDir3pm'] = df['WindDir3pm'].fillna("NaN")
```

In [6]:
```
column_transformer = ColumnTransformer([
    ("windgustdir",OneHotEncoder(), ["WindGustDir"]),
    ("winddir9am",OneHotEncoder(), ["WindDir9am"]),
    ("winddir3pm",OneHotEncoder(), ["WindDir3pm"]),
    ("loc",OneHotEncoder(), ["Location"]),],
    remainder="passthrough")
```

In [7]:
```
#df = df.dropna()
len(df)
```

Out[7]:
```
140787
```

In [8]:
```
x_test = df[df['Year']>=2016]
y_test = x_test.pop('RainTomorrow')
x_train = df[df['Year']<2016]
y_train = x_train.pop('RainTomorrow')
```

In [9]:
```
column_transformer.fit(x_train)
```

Out[9]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('windgustdir', OneHotEncoder(),
                                 ['WindGustDir']),
                                ('winddir9am', OneHotEncoder(), ['Win
dDir9am']),
                                ('winddir3pm', OneHotEncoder(), ['Win
dDir3pm']),
                                ('loc', OneHotEncoder(), ['Location
'])])
```

In [10]:
```
foo = pd.DataFrame(data=column_transformer.transform(x_train).toarray
())
```

First Model

- RandomForestClassifier

In [11]:
```
# rfc = make_pipeline(column_transformer,
#                     FunctionTransformer(lambda x: x.todense(), accept
_sparse=True),
#                     MinMaxScaler(),
#                     KNNImputer(),
#                     RandomForestClassifier())
# rfc.fit(x_train, y_train)
# rfc.score(x_test, y_test)


# Score is 0.8453908984830805
```

In [12]:
```
rfc = make_pipeline(column_transformer,
                    MaxAbsScaler(),
                    SimpleImputer(),
                    KNeighborsClassifier())
rfc.fit(x_train, y_train)
rfc.score(x_test, y_test)
#rfc=RandomForestClassifier()
```

Out[12]:
```
0.7945546479968884
```

In [18]:
```
classifier_dict = {}

for classifier in [
    LogisticRegression(),
    SGDClassifier(),
    Perceptron(),
    PassiveAggressiveClassifier(),
    #NuSVC(nu=0.2),
    LinearSVC(),
    SVC(),
    KNeighborsClassifier(),
    #RadiusNeighborsClassifier(radius=1.5), # Nothing found in radius,
even after increase
    NearestCentroid(),
    #MultinomialNB(), Negative values?
    BernoulliNB(),
    DecisionTreeClassifier(),
    BaggingClassifier(),
    RandomForestClassifier(),
    ExtraTreesClassifier(),
    AdaBoostClassifier(),
    MLPClassifier(),
]:
    start = time.time()
    pipeline = make_pipeline(column_transformer,
                    MaxAbsScaler(),
                    SimpleImputer(),
                    classifier)
    pipeline.fit(x_train, y_train)
    score = pipeline.score(x_test, y_test)
    training_time = time.time() - start
    print(f"{type(classifier).__name__} ({training_time} seconds): {sco
re}")
    classifier_dict[type(classifier).__name__] = {"classifier": pipelin
e, "score": score, "training_time": training_time}

pprint.pprint(classifier_dict)
```

```
BernoulliNB (0.5497801303863525 seconds): 0.7530143912874367
DecisionTreeClassifier (21.280158281326294 seconds): 0.773123298327499
BaggingClassifier (150.1186318397522 seconds): 0.8358615324776352
RandomForestClassifier (223.84542393684387 seconds): 0.8471022948269156
ExtraTreesClassifier (439.9449441432953 seconds): 0.847024504084014
AdaBoostClassifier (8.627912521362305 seconds): 0.8387008945935434

C:\Users\steve\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier (1029.1785008907318 seconds): 0.8369117075068067
{'AdaBoostClassifier': {'classifier': Pipeline(steps=[('columntransfo
rmer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('windgustdir',
                                                  OneHotEncoder(),
                                                  ['WindGustDir']),
                                                 ('winddir9am', OneHo
tEncoder(),
                                                  ['WindDir9am']),
                                                 ('winddir3pm', OneHo
tEncoder(),
                                                  ['WindDir3pm']),
                                                 ('loc', OneHotEncode
r(),
                                                  ['Location'])])),
                ('maxabsscaler', MaxAbsScaler()),
                ('simpleimputer', SimpleImputer()),
                ('adaboostclassifier', AdaBoostClassifier())]),
                       'score': 0.8387008945935434,
                       'training_time': 8.627912521362305},
 'BaggingClassifier': {'classifier': Pipeline(steps=[('columntransfor
mer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('windgustdir',
                                                  OneHotEncoder(),
                                                  ['WindGustDir']),
                                                 ('winddir9am', OneHo
tEncoder(),
                                                  ['WindDir9am']),
                                                 ('winddir3pm', OneHo
tEncoder(),
                                                  ['WindDir3pm']),
                                                 ('loc', OneHotEncode
r(),
                                                  ['Location'])])),
                ('maxabsscaler', MaxAbsScaler()),
                ('simpleimputer', SimpleImputer()),
                ('baggingclassifier', BaggingClassifier())]),
                       'score': 0.8358615324776352,
                       'training_time': 150.1186318397522},
 'BernoulliNB': {'classifier': Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('windgustdir',
                                                  OneHotEncoder(),
                                                  ['WindGustDir']),
                                                 ('winddir9am', OneHo
tEncoder(),
                                                  ['WindDir9am']),
                                                 ('winddir3pm', OneHo
tEncoder(),
                                                  ['WindDir3pm']),
                                                 ('loc', OneHotEncode
r(),
                                                  ['Location'])])),
                ('maxabsscaler', MaxAbsScaler()),
```

```
                ('simpleimputer', SimpleImputer()),
                ('bernoullinb', BernoulliNB())]),
                 'score': 0.7530143912874367,
                 'training_time': 0.5497801303863525},
 'DecisionTreeClassifier': {'classifier': Pipeline(steps=[('columntra
nsformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('windgustdir',
                                                  OneHotEncoder(),
                                                  ['WindGustDir']),
                                                 ('winddir9am', OneHo
tEncoder(),
                                                  ['WindDir9am']),
                                                 ('winddir3pm', OneHo
tEncoder(),
                                                  ['WindDir3pm']),
                                                 ('loc', OneHotEncode
r(),
                                                  ['Location']),])),
                ('maxabsscaler', MaxAbsScaler()),
                ('simpleimputer', SimpleImputer()),
                ('decisiontreeclassifier', DecisionTreeClassifier
())]),
                           'score': 0.773123298327499,
                           'training_time': 21.280158281326294},
 'ExtraTreesClassifier': {'classifier': Pipeline(steps=[('columntrans
former',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('windgustdir',
                                                  OneHotEncoder(),
                                                  ['WindGustDir']),
                                                 ('winddir9am', OneHo
tEncoder(),
                                                  ['WindDir9am']),
                                                 ('winddir3pm', OneHo
tEncoder(),
                                                  ['WindDir3pm']),
                                                 ('loc', OneHotEncode
r(),
                                                  ['Location']),])),
                ('maxabsscaler', MaxAbsScaler()),
                ('simpleimputer', SimpleImputer()),
                ('extratreesclassifier', ExtraTreesClassifier())]),
                         'score': 0.847024504084014,
                         'training_time': 439.9449441432953},
```

In [20]:
```
classifier_dict = {}

for classifier in [
    MLPClassifier(activation="tanh", solver="lbfgs", max_iter=2000),
]:
    start = time.time()
    pipeline = make_pipeline(column_transformer,
                    MaxAbsScaler(),
                    SimpleImputer(),
                    classifier)
    pipeline.fit(x_train, y_train)
    score = pipeline.score(x_test, y_test)
    raining_time = time.time() - start
    print(f"{type(classifier).__name__} ({training_time} seconds): {sco
re}")
    classifier_dict[type(classifier).__name__] = {"classifier": pipelin
e, "score": score, "training_time": training_time}

pprint.pprint(classifier_dict)
```

```
C:\Users\steve\anaconda3\lib\site-packages\sklearn\neural_network\_mu
ltilayer_perceptron.py:500: ConvergenceWarning: lbfgs failed to conve
rge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sho
wn in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_it
er)

MLPClassifier (1703.0184400081635 seconds): 0.8081680280046675
{'MLPClassifier': {'classifier': Pipeline(steps=[('columntransformer
',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('windgustdir',
                                                  OneHotEncoder(),
                                                  ['WindGustDir']),
                                                 ('winddir9am', OneHo
tEncoder(),
                                                  ['WindDir9am']),
                                                 ('winddir3pm', OneHo
tEncoder(),
                                                  ['WindDir3pm']),
                                                 ('loc', OneHotEncode
r(),
                                                  ['Location']),])),
                ('maxabsscaler', MaxAbsScaler()),
                ('simpleimputer', SimpleImputer()),
                ('mlpclassifier',
                 MLPClassifier(activation='tanh', max_iter=2000,
                               solver='lbfgs'))]),
   'score': 0.8081680280046675,
   'training_time': 1703.0184400081635}}
```

In [ ]:
```
classifier_dict["RandomForestClassifier"]
```

Modeling

- LogisticRegression

In [ ]:
```
lrc = make_pipeline(column_transformer, LogisticRegression(random_state
=0))
lrc.fit(x_train, y_train)
```

In [ ]:
```
lrc.score(x_test, y_test)
```

- DecisionTreeClassifier

In [ ]:
```
drc = make_pipeline(column_transformer, DecisionTreeClassifier(random_s
tate=0))
drc.fit(x_train, y_train)
```

In [ ]:
```
drc.score(x_test, y_test)
```

Evaluation

Conclusions

Linear Model

Linear Model Feature Engineering

Linear Models