

02. decision_tree

October 8, 2024

1 Predictive Modeling

```
[12]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import httpimport
import joblib

from imblearn.over_sampling import SMOTE
from pathlib import Path
from sklearn import tree
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split, RandomizedSearchCV, \
    GridSearchCV
from sklearn.tree import DecisionTreeClassifier
```

```
[13]: # Import personal library
with httpimport.github_repo("junclemente", "jcds", ref="master"):
    import jcds.metrics as jm
```

```
[14]: # Import datasets
datasets = Path("../datasets")
train_data = "training_data.csv"
val_data = "validation_data.csv"
test_data = "testing_data.csv"
train_df = pd.read_csv(datasets / train_data)
val_df = pd.read_csv(datasets / val_data)
test_df = pd.read_csv(datasets / test_data)
display(train_df.head())
display(val_df.head())
display(test_df.head())
```

	Undergrad_Degree	Work_Experience	Employability_Before	Status	\
0	Computer Science	No	185.174286	Placed	
1	Engineering	No	206.867959	Not Placed	
2	Art	No	234.881837	Not Placed	

3	Finance	No	173.900408	Placed
4	Art	No	184.063980	Not Placed

	Status_enc
0	1
1	0
2	0
3	1
4	0

	Undergrad_Degree	Work_Experience	Employability_Before	Status \
0	Business	Yes	261.272959	Placed
1	Engineering	No	173.558776	Not Placed
2	Finance	No	205.074388	Placed
3	Business	Yes	230.526020	Placed
4	Business	No	229.000000	Not Placed

	Status_enc
0	1
1	0
2	1
3	1
4	0

	Undergrad_Degree	Work_Experience	Employability_Before	Status \
0	Finance	No	168.775918	Placed
1	Business	Yes	195.508673	Placed
2	Computer Science	No	260.760510	Placed
3	Art	No	231.892551	Not Placed
4	Computer Science	Yes	400.000000	Placed

	Status_enc
0	1
1	1
2	1
3	0
4	1

1.1 Setup Training and Validation dataframes

```
[15]: # Variables to use for predictive modeling
variables = ["Undergrad_Degree", "Work_Experience", "Employability_Before"]
target = "Status_enc"
```

```
[16]: # Setup train and val dataframes
X_train = train_df[variables]
y_train = train_df[target]
X_val = val_df[variables]
```

```

y_val = val_df[target]
X_test = test_df[variables]
y_test = test_df[target]

# One-hot encode categorical variables
X_train = pd.get_dummies(X_train, drop_first=True)
X_val = pd.get_dummies(X_val, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)
display(X_train.head())
display(X_val.head())
display(X_test.head())

```

	Employability_Before	Undergrad_Degree_Business \
0	185.174286	False
1	206.867959	False
2	234.881837	False
3	173.900408	False
4	184.063980	False

	Undergrad_Degree_Computer Science	Undergrad_Degree_Engineering \
0	True	False
1	False	True
2	False	False
3	False	False
4	False	False

	Undergrad_Degree_Finance	Work_Experience_Yes
0	False	False
1	False	False
2	False	False
3	True	False
4	False	False

	Employability_Before	Undergrad_Degree_Business \
0	261.272959	True
1	173.558776	False
2	205.074388	False
3	230.526020	True
4	229.000000	True

	Undergrad_Degree_Computer Science	Undergrad_Degree_Engineering \
0	False	False
1	False	True
2	False	False
3	False	False
4	False	False

	Undergrad_Degree_Finance	Work_Experience_Yes
--	--------------------------	---------------------

0	False	True
1	False	False
2	True	False
3	False	True
4	False	False

	Employability_Before	Undergrad_Degree_Business	\
0	168.775918	False	
1	195.508673	True	
2	260.760510	False	
3	231.892551	False	
4	400.000000	False	

	Undergrad_Degree_Computer Science	Undergrad_Degree_Engineering	\
0	False	False	
1	False	False	
2	True	False	
3	False	False	
4	True	False	

	Undergrad_Degree_Finance	Work_Experience_Yes
0	True	False
1	False	True
2	False	False
3	False	False
4	False	True

1.1.1 Check class balance

```
[17]: y_train.value_counts()
```

```
[17]: Status_enc
1    348
0    228
Name: count, dtype: int64
```

```
[24]: # Use SMOTE to balance classes
smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)

y_train.value_counts()
```

```
[24]: Status_enc
1    348
0    348
Name: count, dtype: int64
```

2 Decision Tree

2.1 RandomSearchCV

```
[25]: tree = DecisionTreeClassifier()
param_dist = {
    "criterion": ["gini", "entropy"],
    "max_depth": [None, 10, 20, 30, 40],
    "min_samples_split": [2, 5, 10, 15],
    "min_samples_leaf": [1, 2, 4, 6],
    "max_features": [None, "sqrt", "log2"],
}

random_search = RandomizedSearchCV(
    estimator=tree,
    param_distributions=param_dist,
    n_iter=50,
    cv=5,
    scoring="accuracy",
    random_state=42,
)

random_search.fit(X_train, y_train)

print(random_search.best_params_)

{'min_samples_split': 15, 'min_samples_leaf': 2, 'max_features': None,
 'max_depth': 10, 'criterion': 'gini'}
```

2.2 GridSearchCV

```
[26]: param_grid = {
    "min_samples_split": [10, 15, 20],
    "min_samples_leaf": [1, 2, 3],
    "max_features": [None, "sqrt", "log2"],
    "max_depth": [4, 6, 8, 10],
    "criterion": ["gini", "entropy"],
}

grid_search = GridSearchCV(
    estimator=tree, param_grid=param_grid, cv=5, scoring="accuracy"
)

grid_search.fit(X_train, y_train)
print(grid_search.best_params_)

{'criterion': 'entropy', 'max_depth': 6, 'max_features': None,
 'min_samples_leaf': 2, 'min_samples_split': 10}
```

2.3 Prediction Model

```
[27]: dt_model = DecisionTreeClassifier(  
        criterion="entropy",  
        max_depth=6,  
        max_features=None,  
        min_samples_leaf=2,  
        min_samples_split=10,  
        random_state=42,  
    )  
    dt_model.fit(X_train, y_train)  
  
    y_pred = dt_model.predict(X_val)
```

```
[28]: cm = confusion_matrix(y_val, y_pred)  
    jm.mc_confusion(cm)
```

Confusion Matrix:

```
[[154   7]  
 [  5 218]]
```

```
[28]:
```

	Class 0	Class 1
Accuracy	0.96875	0.96875
Error rate	0.03125	0.03125
Sensitivity (Recall)	0.95652	0.97758
Specificity	0.97758	0.95652
Precision	0.96855	0.96889
F1	0.96250	0.97321
F2	0.95890	0.97583
F0.5	0.96612	0.97061

2.4 Test

```
[29]: test_pred = dt_model.predict(X_test)  
    cm_test = confusion_matrix(y_test, test_pred)  
    jm.mc_confusion(cm_test)
```

Confusion Matrix:

```
[[ 93   2]  
 [  6 139]]
```

```
[29]:
```

	Class 0	Class 1
Accuracy	0.96667	0.96667
Error rate	0.03333	0.03333
Sensitivity (Recall)	0.97895	0.95862
Specificity	0.95862	0.97895
Precision	0.93939	0.98582
F1	0.95876	0.97203
F2	0.97077	0.96394

F0.5 0.94705 0.98025

3 Export Model

```
[30]: models = Path("../models")  
      joblib.dump(dt_model, models / "decision_tree_model.pkl")
```

```
[30]: ['../models/decision_tree_model.pkl']
```

```
[ ]:
```