

2.04. logistic_regression

October 14, 2024

1 Predictive Modeling

```
[3]: 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.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split, RandomizedSearchCV, \
    GridSearchCV
from sklearn.preprocessing import StandardScaler
```

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

```
[5]: # 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

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

```
[7]: # Setup train, val, and test 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)

# Standardize cont / Initialize scaler
scaler = StandardScaler()
std_cols = ["Employability_Before"]
X_train[std_cols] = scaler.fit_transform(X_train[std_cols])
X_val[std_cols] = scaler.transform(X_val[std_cols])
X_test[std_cols] = scaler.transform(X_test[std_cols])

display(X_train.head())
display(X_val.head())
display(X_test.head())

```

	Employability_Before	Undergrad_Degree_Business \
0	-0.800730	False
1	-0.244034	False
2	0.474848	False
3	-1.090036	False
4	-0.829222	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	1.152088	True
1	-1.098803	False
2	-0.290060	False
3	0.363071	True

4	0.323911	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	-1.221539	False
1	-0.535532	True
2	1.138938	False
3	0.398138	False
4	4.712054	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

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

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

```
[9]: # 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()
```

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

2 Logistic Regression

2.1 RandomSearchCV

```
[15]: log_reg = LogisticRegression()
param_dist = {
    "C": [0.001, 0.01, 0.1, 1, 10],
    "penalty": ["l1", "l2"],
    "solver": ["saga", "liblinear"],
    "max_iter": [100, 200, 250, 300, 400],
}

random_search = RandomizedSearchCV(
    estimator=log_reg,
    param_distributions=param_dist,
    n_iter=10,
    cv=5,
    scoring="accuracy",
    random_state=42,
)

random_search.fit(X_train, y_train)

print(random_search.best_params_)
```

```
{'solver': 'liblinear', 'penalty': 'l2', 'max_iter': 100, 'C': 10}
```

```
/home/nobody/miniconda3/envs/ads505fp/lib/python3.8/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
/home/nobody/miniconda3/envs/ads505fp/lib/python3.8/site-
packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was
reached which means the coef_ did not converge
  warnings.warn(
```

2.2 GridSearchCV

```
[16]: param_grid = {
        "C": [0.001, 0.005, 0.01, 0.05, 0.1, 1],
        "penalty": ["l1", "l2"],
        "solver": ["saga", "liblinear"],
        "max_iter": [100, 200, 250, 300, 400],
    }

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

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

{'C': 0.05, 'max_iter': 100, 'penalty': 'l1', 'solver': 'liblinear'}
```

2.3 Prediction Model

```
[17]: lr_model = LogisticRegression(C=0.05, max_iter=100, penalty="l1",
    ↪ solver="liblinear")
    lr_model.fit(X_train, y_train)

    y_pred = lr_model.predict(X_val)
```

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

Confusion Matrix:

```
[[153   8]
 [  5 218]]
```

```
[18]:
```

	Class 0	Class 1
Accuracy	0.96615	0.96615
Error rate	0.03385	0.03385
Sensitivity (Recall)	0.95031	0.97758
Specificity	0.97758	0.95031
Precision	0.96835	0.96460
F1	0.95925	0.97105
F2	0.95387	0.97496
F0.5	0.96469	0.96717

2.4 Test

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

Confusion Matrix:

```
[[ 91   4]
 [  3 142]]
```

[19]:		Class 0	Class 1
	Accuracy	0.97083	0.97083
	Error rate	0.02917	0.02917
	Sensitivity (Recall)	0.95789	0.97931
	Specificity	0.97931	0.95789
	Precision	0.96809	0.97260
	F1	0.96296	0.97595
	F2	0.95992	0.97796
	F0.5	0.96603	0.97394

3 Export Model

```
[20]: models = Path("../models")
      joblib.dump(lr_model, models / "logistic_regression_model.pkl")
```

```
[20]: ['../models/logistic_regression_model.pkl']
```

```
[ ]:
```