Week 9: Brandon Mather DSC550-T301

1) Import the dataset and ensure that it loaded properly.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
train = pd.read_csv("Loan_Train.csv")
train
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

2) Prepare the data for modeling by performing the following steps:

Drop the column “Load_ID.”

Drop any rows with missing data.

Convert the categorical features into dummy variables.

In [3]:
```
train.drop(columns=['Loan_ID'])
```

Out[3]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.0
1	Male	Yes	1	Graduate	No	4583	1508.0
2	Male	Yes	0	Graduate	Yes	3000	0.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0
4	Male	No	0	Graduate	No	6000	0.0
...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0
610	Male	Yes	3+	Graduate	No	4106	0.0
611	Male	Yes	1	Graduate	No	8072	240.0
612	Male	Yes	2	Graduate	No	7583	0.0
613	Female	No	0	Graduate	Yes	4583	0.0

614 rows × 12 columns

In [5]:
```
train.dropna(axis=0, inplace = True)
```

In [6]:
```
train = pd.get_dummies(train)
train
```

Out[6]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_ID_
1	4583	1508.0	128.0	360.0	1.0	
2	3000	0.0	66.0	360.0	1.0	
3	2583	2358.0	120.0	360.0	1.0	
4	6000	0.0	141.0	360.0	1.0	
5	5417	4196.0	267.0	360.0	1.0	
...	...	...	...	...	...	
609	2900	0.0	71.0	360.0	1.0	
610	4106	0.0	40.0	180.0	1.0	
611	8072	240.0	253.0	360.0	1.0	
612	7583	0.0	187.0	360.0	1.0	
613	4583	0.0	133.0	360.0	0.0	

480 rows × 502 columns

3)Split the data into a training and test set, where the “Loan_Status” column is the target.

In [7]:
```
train.drop(columns=['Loan_Status_N'])
```

Out[7]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_ID_
1	4583	1508.0	128.0	360.0	1.0	
2	3000	0.0	66.0	360.0	1.0	
3	2583	2358.0	120.0	360.0	1.0	
4	6000	0.0	141.0	360.0	1.0	
5	5417	4196.0	267.0	360.0	1.0	
...	...	...	...	...	...	
609	2900	0.0	71.0	360.0	1.0	
610	4106	0.0	40.0	180.0	1.0	
611	8072	240.0	253.0	360.0	1.0	
612	7583	0.0	187.0	360.0	1.0	
613	4583	0.0	133.0	360.0	0.0	

480 rows × 501 columns

In [8]:
```
x = train.drop('Loan_Status_Y',axis=1)
y = train.Loan_Status_Y
```

In [9]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

4)Create a pipeline with a min-max scaler and a KNN classifier (see section 15.3 in the Machine Learning with Python Cookbook).

In [10]:
```
scaler = MinMaxScaler()
```

In [11]:
```
x_train_scaled = scaler.fit_transform(x_train)
```

In [12]:
```
x_test_scaled = scaler.transform(x_test)
```

5)Fit a default KNN classifier to the data with this pipeline. Report the model accuracy on the test set. Note: Fitting a pipeline model works just like fitting a regular model.

In [13]:
```
knn = KNeighborsClassifier(n_neighbors = 5)
```

In [14]:
```
knn.fit(x_train_scaled, y_train)
```

Out[14]:
```
KNeighborsClassifier()
```

In [15]:
```
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(x_train_scaled, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(x_test_scaled, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.95
Accuracy of K-NN classifier on test set: 0.89
```

6)Create a search space for your KNN classifier where your “n_neighbors” parameter varies from 1 to 10. (see section 15.3 in the Machine Learning with Python Cookbook).

In [16]:
```
search_space = [{"knn__n_neighbors": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}]
```

7)Fit a grid search with your pipeline, search space, and 5-fold cross-validation to find the best value for the “n_neighbors” parameter.

In [17]:
```
knn2 = KNeighborsClassifier()
```

In [18]:
```
standardizer = StandardScaler()
```

In [19]:
```
pipe = Pipeline([("standardizer", standardizer), ("knn", knn2)])
```

In [20]:
```
knn_gscv = GridSearchCV(pipe, search_space, cv=5)
```

In [21]:
```
knn_gscv.fit(x, y)
```

Out[21]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('standardizer', StandardScaler()),
                                       ('knn', KNeighborsClassifier())]),
             param_grid=[{'knn__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}])
```

In [22]:
```
knn_gscv.best_params_
```

Out[22]:
```
{'knn__n_neighbors': 4}
```

8)Find the accuracy of the grid search best model on the test set. Note: It is possible that this will not be an improvement over the default model, but likely it will be.

In [23]:
```
knn_gscv.best_score_
```

Out[23]:
```
0.9020833333333333
```

9)Now, repeat steps 6 and 7 with the same pipeline, but expand your search space to include logistic regression and random forest models with the hyperparameter values in section 12.3 of the Machine Learning with Python Cookbook.

In [24]:
```
search_space2 = [{"classifier": [LogisticRegression()],
"classifier__penalty": ['l2'],
"classifier__C": np.logspace(0, 4, 10)},
{"classifier": [RandomForestClassifier()],
"classifier__n_estimators": [10, 100, 1000],
"classifier__max_features": [1, 2, 3]}]
```

In [25]:
```
pipe2 = Pipeline([("standardizer", standardizer), ("classifier", knn2)])
```

In [26]:
```
gridsearch = GridSearchCV(pipe2, search_space2, cv=5)
```

In [27]:
```
best_model = gridsearch.fit(x, y)
```

10)What are the best model and hyperparameters found in the grid search? Find the accuracy of this model on the test set.

In [28]:
```
best_model.best_estimator_.get_params()["classifier"]
```

Out[28]:
```
LogisticRegression(C=166.81005372000593)
```

In [29]:
```
best_model.best_score_
```

Out[29]:
```
0.9979166666666666
```

11)Summarize your results.

The best model is a Logistic Regression model with 99.7% accuracy.