Model evaluation

In [1]:

```
# Load packages
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
```

In [2]:

```
# Load data
df = pd.read_csv('default.csv')
```

In [3]:

```
# Split data in training and test set
df_train, df_test = train_test_split(df, test_size=0.4, stratify=df.default,
                                     random_state=24)
```

In [4]:

```
# Prepare training data
X_train = pd.get_dummies(df_train, columns=['student'], drop_first=True).drop('default'
, axis=1)
y_train = df_train['default']
```

In [5]:

```
# Fit logistic regression model on training set
model = LogisticRegression(C=1e9, tol=1e-5)
model.fit(X_train.values, y_train.values)
```

Out[5]:

```
LogisticRegression(C=1000000000.0, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=1e-05, verbose=0, warm_start=False)
```

In [6]:

```
# Prepare test set
X_test = pd.get_dummies(df_test, columns=['student'], drop_first=True).drop('default',
axis=1)
y_test = df_test['default']
```

In [7]:

```
# Get probabilities for test set
probabilities_test = model.predict_proba(X_test.values)
probabilities_test[:5, 1].round(3)
```

Out[7]:

```
array([0.   , 1.   , 0.001, 0.999, 0.983])
```

In [8]:

```
# Predict class for test set
predictions_test = model.predict(X_test.values)
predictions_test[:5]
```

Out[8]:

```
array([0, 1, 0, 1, 1], dtype=int64)
```

In [9]:

```
# Print confusion matrix
confusion_matrix(y_test, predictions_test)
```

Out[9]:

```
array([[19,  1],
       [ 2, 18]], dtype=int64)
```

In [10]:

```
# Store elements of confusion matrix
tn, fp, fn, tp = confusion_matrix(y_test, predictions_test).ravel()
```

In [11]:

```
# Display accuracy
accuracy_score(y_test, predictions_test)
```

Out[11]:

```
0.925
```

In [12]:

```
# Display precision
precision_score(y_test, predictions_test)
```

Out[12]:

```
0.9473684210526315
```

In [13]:

```
# Display true positive rate (recall)
recall_score(y_test, predictions_test)
```

Out[13]:

```
0.9
```

In [14]:

```
# Display false positive rate
fp / (fp + tn)
```

Out[14]:

```
0.05
```

In [15]:

```
# Display F1-score
f1_score(y_test, predictions_test)
```

Out[15]:

```
0.9230769230769231
```

In [16]:

```
# Display AUC
roc_auc_score(y_test, probabilities_test[:, 1])
```

Out[16]:

```
0.9750000000000001
```