Classification Using Logistic Regression

Concept Session

Demo 3.1: Data Preparation

Import Data & Python Packages

In [ ]:
```
#first we have to import all relevant python packages
import numpy as np
import pandas as pd

from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import matplotlib.pyplot as plt
from matplotlib import pyplot
plt.rc("font", size=14)
import seaborn as sns
sns.set(style="white") #white background style for seaborn plots
sns.set(style="whitegrid", color_codes=True)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

import warnings
warnings.simplefilter(action='ignore')
```

In [ ]:
```
# mount the drive
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
# Read CSV train data file into DataFrame
loan_df = pd.read_csv('/content/drive/MyDrive/ML/DS2_C5_S3_Loan_Data_Concept.csv')

# preview the data
loan_df.head()
```

In [ ]:
```
# shape of the dataset
print('The number of samples in data is {}.'.format(loan_df.shape[0]))
```

1. Data Exploration and Visualization - Understanding the data

In [ ]:
```
# explore the existing data types
loan_df.dtypes
```

In [ ]:
```
#explore the numeric data types
loan_df.describe()
```

In [ ]:
```
# explore the strings
loan_string = loan_df.select_dtypes(exclude=[np.number])
loan_string.head(3)
```

It seems like we have duplicates in our data. Therefore, we will clean our data in the following steps.

Purpose wise not fully paid rate

In [ ]:
```
pd.crosstab(loan_df['purpose'],loan_df['not.fully.paid']).plot(kind='bar')
```

From the above chart, gender seems to play an important role in regards to the people who survived.

2. Data Preprocessing

Data quality | Missing Value Assessment

In [ ]:
```
# check missing values in data
loan_df[loan_df.isnull().any(axis=1)]
```

As we can see, no missing value exists in this dataset

Data quality | Detect correlations

In [ ]:
```
column_correlation = loan_df.corr()
column_correlation
```

In [ ]:
```
import seaborn as sns

sns.heatmap(column_correlation);
plt.show()
```

We are not removing any feature from this dataset, as we can see none of the columns are highly correlated

Preparation of data

The 'purpose' variable is a categorical variable. Therefore, it needs to be converted to numbers.

In [ ]:
```
#Encoding purpose variable to numeric variable
purpose_encoder = preprocessing.LabelEncoder()

# Encoding of the purpose
purpose_encoder.fit(loan_df['purpose'])
print(purpose_encoder.classes_)
loan_df['purpose'] = purpose_encoder.transform(loan_df['purpose'])

loan_df.head()
```

Demo 3.2: Logistic Regression

1. Preparation of training and test dataset

In [ ]:
```
#Creating test and training datasets
loan_train, loan_test = train_test_split(loan_df,train_size = 0.8)

print('Size of training dataset: ', loan_train.shape)
print('Size of test dataset: ', loan_test.shape)
```

In [ ]:
```
X_train = loan_train.drop(columns='not.fully.paid', axis =1)
Y_train = loan_train['not.fully.paid']
X_test = loan_test.drop(columns='not.fully.paid', axis =1)
Y_test = loan_test['not.fully.paid']
X_train.shape, Y_train.shape, X_test.shape
```

Installment vs. not fully paid

In [ ]:
```
plt.scatter(X_test['installment'],Y_test)
plt.xlabel('Installment')
plt.ylabel('not.fully.paid')
plt.show()
```

2. Training the model

In [ ]:
```
# Logistic Regression - training the model
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
# Using the trained model to predict the outcome of the X_Test dataset
Y_pred = logreg.predict(X_test)
#Calculating the accuracy of the training dataset
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

In [ ]:
```
# indicates the value of the slope of each parameter
coeff_df = pd.DataFrame(loan_train.columns.delete(0))
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Correlation', ascending=False)
```

In [ ]:
```
#model.intercept_ indicates the intercept with the Y-axis
logreg.intercept_
```

In [ ]:
```
# evalate the model on the test data
logreg.score(X_test, Y_test)
```

In [ ]:
```
#indicating which person will survive in the dataset
np.unique(Y_pred)
```

In [ ]:
```
from collections import Counter

x = 0
d = Counter(Y_pred)
print('{} has occurred {} times'.format(x, d[x]))
```

Demo 3.3: Model Evaluation

1. Confusion matrix

In [ ]:
```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(Y_test,Y_pred))

tn, fp, fn, tp = confusion_matrix(Y_test,Y_pred).ravel()
print(tn, fp, fn, tp)
```

In [ ]:
```
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score

print('accuracy:', accuracy_score(Y_test, Y_pred))
print('recall:', recall_score(Y_test, Y_pred))
print('f1-score:', f1_score(Y_test, Y_pred))
print('precision:', precision_score(Y_test, Y_pred))
```

2. ROC-AUC

In [ ]:
```
from sklearn.metrics import roc_auc_score
roc=roc_auc_score(Y_test, logreg.predict_proba(X_test)[:,1])
roc
```

In [ ]:
```
# calculate roc curve
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(Y_test, Y_pred)
# calculate AUC
auc = roc_auc_score(Y_test, Y_pred)
print('AUC: %.2f' % auc)
```

In [ ]:
```
# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
```

In [ ]:
```
# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(Y_test))]
```

In [ ]:
```
# predict probabilities
lr_probs = logreg.predict_proba(X_test)
```

In [ ]:
```
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# calculate scores
ns_auc = roc_auc_score(Y_test, ns_probs)
lr_auc = roc_auc_score(Y_test, lr_probs)
# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(Y_test, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(Y_test, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```