```
#
# Purdue University Global
#
# IN402 - Modeling and Predictive Analysis
#
# Unit 10 Assignment / Module 6 Part 3 Competency Assessment
#
# Classification Model Selection
#
# PyCharm Code
#

# Import packages

import sys

import pandas as pd

# Ignoring warnings

if not sys.warnoptions:
    import warnings

warnings.simplefilter("ignore")

# Output Header

print('Unit 10 Assignment / Module 6 Part 3 Competency Assessment Output\n')

from datetime import datetime

print(datetime.now().strftime("%m/%d/%Y %H:%M:%S"), '\n')

# Import the dataset

# Importing the dataset to a pandas DataFrame

df = pd.read_csv('/home/codio/workspace/data/IN402/Churn_Modelling.csv')

print(df.shape)

# Wrangle the data

# Drop columns with no analytical value

df = df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)

# Convert the categorical columns into dummy columns
# and drop the original categorical columns

geography = pd.get_dummies(df.Geography).iloc[:,1:]

gender = pd.get_dummies(df.Gender).iloc[:,1:]

# Drop columns with non-numeric data

df = df.drop(['Geography', 'Gender'], axis=1)

# Join the dummy columns into the main dataset

# Add columns with converted dummy values

df = pd.concat([df,geography,gender], axis= 1)

# Split the dataset into target and feature subsets.

X = df.drop(['Exited'], axis=1)

y = df.loc[:, 'Exited']

# Select features

# Check the variance in the numeric variables

from statistics import variance

creditScore = df['CreditScore']

age = df['Age']

tenure = df['Tenure']

balance = df['Balance']

estimatedSalary = df['EstimatedSalary']

# Display the parameter variances

print("Variance of CreditScore is % s " %(variance(creditScore)))

print("Variance of Age is % s " %(variance(age)))

print("Variance of Tenure is % s " %(variance(tenure)))

print("Variance of Balance is % s " %(variance(balance)))

print("Variance of EstimatedSalary is % s " %(variance(estimatedSalary)))

# Split the dataset into training and testing subsets.

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Display size of training set

print(len(X_train))

# Display size of test set

print(len(X_test))

# Conduct feature scaling (required by SVM)

from sklearn.preprocessing import StandardScaler

# feature scaling is required by SVC

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)

X_train

# Model using Logistic Regression

# Import packages

from sklearn.linear_model import LogisticRegression

from sklearn import metrics

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Build the model

lr_model = LogisticRegression()

# Fit the model

result = lr_model.fit(X_train, y_train)

# Predict using the model

prediction_test = lr_model.predict(X_test)

# Evaluate the model

# Print the prediction accuracy

print(metrics.accuracy_score(y_test, prediction_test))

# Display the confusion matrix

print(confusion_matrix(y_test, prediction_test))

# Display the classification report

print(classification_report(y_test, prediction_test))

# To get the weights of all the variables

weights = pd.Series(lr_model.coef_[0], index=X.columns.values)

weights.sort_values(ascending = False)

# Model using SVM

# Import packages

from sklearn.svm import SVC

from sklearn import metrics

# Build the model

svm_model = SVC(kernel = "linear")

# Fit the model

# Train the model

svm_model.fit(X_train, y_train)

# Predict using the model

svm_prediction = svm_model.predict(X_test)

# Evaluate the model

print("accuracy: ", metrics.accuracy_score(y_test, y_pred = svm_prediction))

# Precision score

print("precision: ", metrics.precision_score(y_test, y_pred = svm_prediction))

# Recall score

print("recall", metrics.recall_score(y_test, y_pred = svm_prediction))

# Display classification report

print(metrics.classification_report(y_test, y_pred = svm_prediction))

# Model using RandomForestClassifier

# Import packages

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, accuracy_score

# Build the model

rf_model = RandomForestClassifier(n_estimators=200, random_state=0)

# Fit the model

rf_model.fit(X_train, y_train)

# Predict using the model

rf_predictions = rf_model.predict(X_test)

# Evaluate the model

print(classification_report(y_test,rf_predictions))

# Display the accuracy score

print(accuracy_score(y_test, rf_predictions ))

# Display the accuracy score

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

feat_importances = pd.Series(rf_model.feature_importances_, index=X.columns)

feat_importances.nlargest(10).plot(title="Accuracy Score", kind='barh')

plt.show()
```