Introduction to Big Data

- Developed by Dr. Keungoui KIM
- https://awekim.github.io/portfolio/

Lecture 11. Classification

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import numpy as np
import pandas as pd
import seaborn as sns
```

Classification with Personal Loan Data

- Experience
- Income
- Famliy
- CCAvg: Average monthly card spent
- Education: Education level (1: undergrad; 2, Graduate; 3; Advance)
- Mortgage
- Securities account: Securities (1:Yes, 0:No)
- CD account: CD account (1:Yes, 0:No)
- Online: Online account (1:Yes, 0:No)
- CreidtCard: Credit Card (1:Yes, 0:No)

```
PerLoan = pd.read_csv("/content/drive/MyDrive/[Lecture]/IntBigData/BigData_Python/11_Classificati
PerLoan.head()
```

```
PerLoan.shape
```

```
PerLoan.columns
```

```
PerLoan.rename(columns={'Personal Loan':'PersonalLoan'},
               inplace=True)
```

```
PerLoan.columns
```

```
PerLoan.describe()
```

```
# check missing values
PerLoan.isnull().any()
```

```
PerLoan.count()
```

```
PL_X = PerLoan[['Age','CCAvg','Income','Education']]
PL_Y = PerLoan['PersonalLoan']
```

Logit Regression with statsmodels

```
import statsmodels.api as sm

statsLogitModel = sm.Logit(PL_Y, PL_X)
statsLogitModel

from statsmodels.formula.api import logit

statsLogitModel = (
    logit('PersonalLoan ~ Age + CCAvg + Income + Education',
          data=PerLoan))
statsLogitModel

statsLogitModel_res = statsLogitModel.fit()

print(statsLogitModel_res.summary())

statsLogitModel_res.params

np.exp(statsLogitModel_res.params)
```

Logit Regression with sklearn

```
# from sklearn import metrics
from sklearn.linear_model import LogisticRegression

LogitModel0 = LogisticRegression()

LogitModel0_res = LogitModel0.fit(PL_X, PL_Y)
LogitModel0_res

LogitModel0_res.coef_

LogitModel0_res.intercept_

from sklearn.model_selection import train_test_split

PL_X_train, PL_X_test, PL_Y_train, PL_Y_test = train_test_split(PL_X, PL_Y, test_size=0.3, random_

# Practice of Random Sampling
PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                    test_size=0.3)
PL_X_train1.shape

PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                    test_size=0.3)
PL_X_train1.head()

PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                    test_size=0.3, random_state=1
PL_X_train1.head()
```

```
PL_X_train1, PL_X_test1, PL_Y_train1, PL_Y_test1 = train_test_split(PL_X, PL_Y,
                                                                    test_size=0.3, random_state=1
PL_X_train1.head()
```

```
PL_X_train
```

```
PL_X_test
```

```
PL_Y_train
```

```
PL_Y_test
```

```
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

LogitModel = LogisticRegression()

LogitModel.fit(PL_X_train, PL_Y_train)

LogitModel.coef_

LogitModel.intercept_

PL_Y_pred = LogitModel.predict(PL_X)

PL_Y_train_pred = LogitModel.predict(PL_X_train)

PL_Y_test_pred = LogitModel.predict(PL_X_test)
```

Validation

```
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, confusion_ma
```

Accuracy Score

```
accuracy_score(PL_Y_test, PL_Y_test_pred)
```

Recall Score

```
recall_score(PL_Y_test, PL_Y_test_pred)
```

Precision Score

```
precision_score(PL_Y_test, PL_Y_test_pred)
```

F1 Score

```
f1_score(PL_Y_test, PL_Y_test_pred)
```

Confusion Matrix

```
confusion_matrix(PL_Y_test, PL_Y_test_pred)
```

Specificity

```
tn, fp, fn, tp = confusion_matrix(PL_Y_test, PL_Y_test_pred).ravel()
specificity = tn / (tn+fp)
specificity
```

ROC Curve

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

logit_roc_auc = roc_auc_score(PL_Y_test, LogitModel.predict(PL_X_test))
fpr, tpr, thresholds = roc_curve(PL_Y_test, LogitModel.predict_proba(PL_X_test)[:,1])

plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

K-means Clustering

```
PL_X_all = PerLoan[['Age','Experience', 'Income', 'Family', 'CCAvg',
            'Education', 'Mortgage', 'Securities Account',
            'CD Account', 'Online', 'CreditCard']]

from sklearn.cluster import KMeans

KMeansModel = KMeans(n_clusters = 2)

KMeansModel.fit(PL_X)

KMeansModel_pred_df = pd.DataFrame(KMeansModel.predict(PL_X),
                                   columns=['KMeansClass'])
KMeansModel_pred_df.head()

KMeansModel_pred_df = pd.concat([PL_X_all, PL_Y, KMeansModel_pred_df], axis=1)
KMeansModel_pred_df.head()

type(KMeansModel_pred_df)
```

```
KMeansModel_pred_df.KMeansClass.unique()

KMeansModel_pred_df[KMeansModel_pred_df.KMeansClass==1].head()

sns.set(rc = {'figure.figsize':(15,8)})
sns.scatterplot(x='CCAvg', y='Age', data=KMeansModel_pred_df,
                hue='KMeansClass')

sns.set(rc = {'figure.figsize':(15,8)})
sns.scatterplot(x='CCAvg', y='Age', data=KMeansModel_pred_df,
                hue='PersonalLoan')

sns.scatterplot(KMeansModel_pred_df['CCAvg'],KMeansModel_pred_df['Age'], hue=KMeansModel_pred_df['K
```

```
sns.scatterplot(x='total_bill',y='tip',data=tips, hue='sex')
```

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn import preprocessing

# 원본 데이터를 복사해서 전처리하기 (원본 데이터를 가지고 바로 전처리하지 않는다)
processed_data = data.copy()

# 데이터 전처리 - 정규화를 위한 작업
scaler = preprocessing.MinMaxScaler()

processed_data[['ItemsBought', 'ItemsReturned']] = scaler.fit_transform(processed_data[['ItemsBough
```

```
KMeansModel.fit(var_X)
```

```
ids = estimator.fit_predict(processed_data[['ItemsBought', 'ItemsReturned']])

# 2행 3열을 가진 서브플롯 추가 (인덱스 = i)

plt.subplot(3, 2, i)
plt.tight_layout()
```

```
# 서브플롯의 라벨링
plt.title("K value = {}".format(i))
plt.xlabel('ItemsBought')
plt.ylabel('ItemsReturned')

# 클러스터링 그리기
plt.scatter(processed_data['ItemsBought'], processed_data['ItemsReturned'], c=ids)
plt.show()
```

Classification with Titanic Data

Data preparation

```
titanic = sns.load_dataset("titanic")

titanic.head()

titanic.shape

titanic.columns

titanic.describe()

titanic['alive_d'] = titanic['alive'].map({'yes':1,'no':0})
titanic['male_d'] = titanic['sex'].map({'male':1,'female':0})
titanic.head()

pd.get_dummies(titanic['sex']).head()

# check missing values
titanic.isnull().any()

titanic.count()

# Find missing values
titanic[titanic['age'].isnull()]
```

```
# Filling missing values
titanic['age'].fillna(value=titanic['age'].mean(), inplace=True)
titanic.isnull().any()

var_X = titanic[['age','pclass','male_d']]
var_Y = titanic['survived']
```

Logit Regression with statsmodels

```
import statsmodels.api as sm

statsLogitModel=sm.Logit(var_Y,var_X)
statsLogitModel

statsLogitModel_res=statsLogitModel.fit()

statsLogitModel_res.summary()
```

Logit Regression with sklearn

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(var_X, var_Y, test_size=0.3, random_state=0)

X_train

X_test

Y_train

Y_test
```

StandardScaler

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logit Regression with sklearn

```
from sklearn.linear_model import LogisticRegression
LogitModel = LogisticRegression()

LogitModel.fit(X_train, Y_train)

LogitModel.coef_
```

```
Y_pred = LogitModel.predict(X_test)
```

Accuracy

```
LogitModel.score(X_test, Y_test)
```

```
LogitModel.score(X_train, Y_train)
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(Y_test, Y_pred)
print(confusion_matrix)

# 150 + 43: correct prediction
# 57 + 18: incorrect prediction
```

Classification report

```
from sklearn.metrics import classification_report

print(classification_report(Y_test, Y_pred))
```

ROC Curve

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

logit_roc_auc = roc_auc_score(Y_test, LogitModel.predict(X_test))
fpr, tpr, thresholds = roc_curve(Y_test, LogitModel.predict_proba(X_test)[:,1])

plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```