**Lab4**

201635825 오성원

**Lab4-1**

|  |
| --- |
| Source Code |
| import pandas as pd  import numpy as np  import io  import warnings  warnings.filterwarnings(action='ignore')  import sklearn.linear\_model as lm  from matplotlib import pyplot as plt  from sklearn.model\_selection import GridSearchCV  from sklearn.model\_selection import KFold  #Read file  df=pd.read\_csv('C:/python\_file/linear\_regression\_data.csv',encoding='utf-8')  #Split the dataset into 4/5 for training and 1/5 for testing.  testset=df.sample(frac=0.2)  trainset=df.drop(testset.index)  training\_distance=trainset['Distance']  training\_time=trainset['Delivery Time']  test\_distance=testset['Distance']  test\_time=testset['Delivery Time']  arr\_training\_distance=np.array(training\_distance)  arr\_training\_time=np.array(training\_time)  arr\_test\_distance=np.array(test\_distance)  arr\_test\_time=np.array(test\_time)  #Creating a model using training data  reg=lm.LinearRegression()  reg.fit(arr\_training\_distance[:, np.newaxis], arr\_training\_time)  #predict delivery time using test diatance value  x=reg.predict(arr\_test\_distance[ : ,np.newaxis])  #Create dataframe for output using test data  df\_test=pd.DataFrame(arr\_test\_distance)  df\_test.rename(columns={df\_test.columns[0]:'Distance'},inplace=True)  df\_test["Delivery Time"]=arr\_test\_time  df\_test["Prediction Delivery Time"]=x  print(df\_test)  print("\n\nK-Flod")  #enumerate splits  count=1  kfold=KFold(n\_splits=5, shuffle=True, random\_state=0)  for train,test in kfold.split(df):  df\_train=df.iloc[train]  df\_train=df\_train['Distance']  df\_train\_t=df.iloc[train]  df\_train\_t=df\_train\_t['Delivery Time']  df\_test=df.iloc[test]  df\_test=df\_test['Distance']  df\_test\_t=df.iloc[test]  df\_test\_t=df\_test\_t['Delivery Time']  #create model  reg.fit(df\_train[:, np.newaxis], df\_train\_t)  param\_grid={'fit\_intercept':['True','False'],'normalize':['True','False'],}  gscv=GridSearchCV(reg,param\_grid,cv=kfold)  #create model  gscv.fit(df\_train[:, np.newaxis],df\_train\_t)  prediction = gscv.predict(df\_test[:,np.newaxis])  result=pd.DataFrame(df\_test)  result.rename(columns={result.columns[0]:'Distance'},inplace=True)  result["Delivery Time"]=df\_test\_t  result["Prediction Delivery Time"]=prediction  print("------------------Result of sample test {0}------------------".format(count))  count+=1  print(result)  print('Best parameter: ',gscv.best\_params\_)  print('Best score: ',gscv.best\_score\_)  print() |

|  |
| --- |
| Screen shot |
|  |

**Lab4-2**

|  |
| --- |
| Source Code |
| import pandas as pd  import numpy as np  import io  import warnings  warnings.filterwarnings(action='ignore')  import matplotlib.pyplot as plt  from sklearn.tree import DecisionTreeClassifier  from sklearn.metrics import accuracy\_score  from sklearn.preprocessing import LabelEncoder  from sklearn.model\_selection import GridSearchCV  from sklearn.model\_selection import KFold  #Read file  df=pd.read\_csv('C:/python\_file/decision\_tree\_data.csv',encoding='utf-8')  #convert data using labelEncoder  labelEncoder = LabelEncoder()  labelEncoder.fit(df['level'])  df['level'] = labelEncoder.transform(df['level'])  labelEncoder.fit(df['lang'])  df['lang'] = labelEncoder.transform(df['lang'])  labelEncoder.fit(df['tweets'])  df['tweets'] = labelEncoder.transform(df['tweets'])  labelEncoder.fit(df['phd'])  df['phd'] = labelEncoder.transform(df['phd'])  #make DecisionTree Model  tree\_model= DecisionTreeClassifier()  count=1  kfold=KFold(n\_splits=10, shuffle=True, random\_state=0)  for train,test in kfold.split(df):  df\_train=df.iloc[train]  train\_x=np.array(df\_train.drop(['interview'],1))  train\_y=np.array(df\_train['interview'])    df\_test=df.iloc[test]  test\_x=np.array(df\_test.drop(['interview'],1))  test\_y=np.array(df\_test['interview'])  param\_grid={'max\_depth' : np.arange(1,10)}  gscv=GridSearchCV(tree\_model, param\_grid, cv=kfold)  #create model  gscv.fit(train\_x,train\_y)  gscv.predict(test\_x)  result\_test=df.iloc[test]  #predict using test\_data  result\_test['Predict']=gscv.predict(test\_x)  print("\n------------------Result of sample test {0}------------------".format(count))  count+=1  print(result\_test)  print('predict best parameters: ',gscv.best\_params\_)  print('predict est score: ',gscv.best\_score\_) |

|  |
| --- |
| Screen shot |
|  |

**Lab4-3**

|  |
| --- |
| Source Code |
| import pandas as pd  import numpy as np  import io  import math  from sklearn.model\_selection import KFold  from sklearn.model\_selection import GridSearchCV  from sklearn.neighbors import KNeighborsClassifier  import warnings  warnings.filterwarnings(action='ignore')  #Read file  df=pd.read\_csv('C:/python\_file/knn\_data.csv',encoding='utf-8')  count=1  #enumerate splits  kfold=KFold(n\_splits=5, shuffle=True, random\_state=0)  for train,test in kfold.split(df):  df\_train=df.iloc[train]  df\_train=df\_train[['longitude','latitude']]  df\_train\_lang=df.iloc[train]  df\_train\_lang=df\_train\_lang['lang']  df\_test=df.iloc[test]  df\_test=df\_test[['longitude','latitude']]  df\_test\_lang=df.iloc[test]  df\_test\_lang=df\_test\_lang['lang']  #create KNN model  knn = KNeighborsClassifier()  param\_grid = {'n\_neighbors': np.arange(1,25)}  knn\_gscv = GridSearchCV (knn, param\_grid , cv=5)  #create model  knn\_gscv.fit(df\_train, df\_train\_lang)    result\_test=df.iloc[test]  #predict using test\_data  result\_test['Predicted lang']=knn\_gscv.predict(df\_test)  print("------------------Result of sample test {0}------------------".format(count))  count+=1  print(result\_test)  print("Predict bet params: {0}".format(knn\_gscv.best\_params\_))  print("Predict best score: {0}\n".format(knn\_gscv.best\_score\_)) |

|  |
| --- |
| Screen shot |
|  |