```
import matplotlib.pyplot as plt
import numpy as np  # data calculate
import pandas as pd  # data analyse
import seaborn as sns

pd.set_option('display.max_colwidth', 1000)
pd.set_option('display.max_rows', None)

data_train = pd.read_csv('/Users/Wenqing Shi/Desktop/CIS5570project/CreditRiskTrain.csv')

print('train data is\n', data_train.head(5), '------aaa\n')

data_train.info()  # there are some missing values in Saving account, Checking account
print('train data describe\n', data_train.describe())

interval = (18, 25, 35, 60, 120)
agelables = ['Student', 'Young', 'Adult', 'Senior']
data_train["Age_lables"] = pd.cut(data_train.Age, interval, labels=agelables)
#
# print(data_train["Saving_accounts"].unique())
data_train["Saving_accounts"] = data_train["Saving_accounts"].fillna('moderate')

# print(data_train["Checking_accounts"].unique())
data_train["Checking_accounts"] = data_train["Checking_accounts"].fillna('moderate')

# print(data_train.head(5))

# data_train.info()  #there are some missing values in Saving account, Checking account
# print(data_train.describe())

# Purpose to Dummies Variable
dummy_Age = pd.get_dummies(data_train.Age_lables, drop_first=False, prefix='Age_lables')
dummy_Sex = pd.get_dummies(data_train.Sex, drop_first=False, prefix='Sex')
dummy_Job = pd.get_dummies(data_train.Job, drop_first=False, prefix='Job')
dummy_Housing = pd.get_dummies(data_train.Housing, drop_first=False, prefix='Housing')
dummy_Saving = pd.get_dummies(data_train.Saving_accounts, drop_first=False,
prefix='Saving_accounts')
dummy_Checking = pd.get_dummies(data_train.Checking_accounts, drop_first=False,
prefix='Checking_accounts')
dummy_Purpose = pd.get_dummies(data_train.Purpose, drop_first=False, prefix='Purpose')
# dummy_Risk = pd.get_dummies(data_train.Risk, drop_first=False, prefix='Risk')
df = pd.concat(
    [data_train, dummy_Age, dummy_Sex, dummy_Job, dummy_Housing, dummy_Saving,
dummy_Checking, dummy_Purpose], axis=1)
```

```
df.drop(['Age', 'Age_lables', 'Job', 'Sex', 'Housing', 'Saving_accounts', 'Checking_accounts',
'Purpose'], axis=1,
        inplace=True)
# Excluding the missing columns


import sklearn.preprocessing as preprocessing

scaler = preprocessing.StandardScaler()
# print('kkkkk',df['Credit_amount'])
std_CreditAmount = scaler.fit_transform(np.reshape(np.array(df['Credit_amount']), (-1, 1)))
std_CreditAmount = pd.DataFrame({'stdCredit_amount': std_CreditAmount[:, 0]})
# print('ddddddd',std_CreditAmount)
df_train = pd.concat([df, std_CreditAmount], axis=1)
df_train.drop(['Credit_amount'], axis=1, inplace=True)

print(df_train.head(5))
df_train.info()  # there are some missing values in Saving account, Checking account
print(df_train.describe())

df_corr = df_train.corr()
f, ax = plt.subplots(figsize=(14, 8))
sns.heatmap(df_train.corr(), linewidths=0.8, vmax=1.0, square=True, linecolor='white',
annot=True)

# plt.show()


from sklearn import linear_model

# 用正则取出我们要的属性值
train_df = df_train.filter(

regex='Risk|Age_.*|Sex_.*|Job_.*|Housing_.*|Saving_accounts_.*|Checking_accounts_.*|std
Credit_amount|Duration|Purpose_.*')
train_np = train_df
print(train_df.head())
# y 即 Risk 结果
y = train_np["Risk"]

# X 即特征属性值
X = train_np.drop("Risk", axis=1)
```

```
clf = linear_model.LogisticRegression(C=1.0, penalty='l1', tol=1e-6)
clf.fit(X, y)

print(clf)

#print ('------\n', cross_validation.cross_val_score(clf, X, y, cv=5))

data_test = pd.read_csv('/Users/Wenqing Shi/Desktop/CIS5570project/CreditRiskTest.csv')
# 接着我们对 test_data 做和 train_data 中一致的特征变换

interval = (18, 25, 35, 60, 120)
agelables = ['Student', 'Young', 'Adult', 'Senior']
data_test["Age_lables"] = pd.cut(data_test.Age, interval, labels=agelables)

data_test['Saving_accounts'] = data_test['Saving_accounts'].fillna('moderate')

data_test['Checking_accounts'] = data_test['Checking_accounts'].fillna('moderate')

# Purpose to Dummies Variable
dummy_Age = pd.get_dummies(data_test.Age_lables, drop_first=False, prefix='Age_lables')
dummy_Sex = pd.get_dummies(data_test.Sex, drop_first=False, prefix='Sex')
dummy_Job = pd.get_dummies(data_test.Job, drop_first=False, prefix='Job')
dummy_Housing = pd.get_dummies(data_test.Housing, drop_first=False, prefix='Housing')
dummy_Saving = pd.get_dummies(data_test.Saving_accounts, drop_first=False,
prefix='Saving_accounts')
dummy_Checking = pd.get_dummies(data_test.Checking_accounts, drop_first=False,
prefix='Checking_accounts')
dummy_Purpose = pd.get_dummies(data_test.Purpose, drop_first=False, prefix='Purpose')
df_test = pd.concat(
   [data_test, dummy_Age, dummy_Sex, dummy_Job, dummy_Housing, dummy_Saving,
dummy_Checking, dummy_Purpose], axis=1)
df_test.drop(['Age', 'Age_lables', 'Job', 'Sex', 'Housing', 'Saving_accounts', 'Checking_accounts',
'Purpose'], axis=1,
            inplace=True)
# Excluding the missing columns
#
std_CreditAmount = scaler.fit_transform(np.reshape(np.array(df_test['Credit_amount']), (-1,
1)))
std_CreditAmount = pd.DataFrame({'stdCredit_amount': std_CreditAmount[:, 0]})
df_test = pd.concat([df_test, std_CreditAmount], axis=1)
df_test.drop(['Credit_amount'], axis=1, inplace=True)

print('df_test is\n', df_test.head(5))
```

```
test = df_test.filter(

regex='Age_.*|Sex_.*|Job_.*|Housing_.*|Saving_accounts_.*|Checking_accounts_.*|stdCredit_amount|Duration|Purpose_.*')
predictions = clf.predict(test)
result = pd.DataFrame({'ID': data_test['ID'].as_matrix(), 'Risk': predictions.astype(np.int32)})
result.to_csv('/Users/Wenqing Shi/Desktop/CIS5570project/logistic_regression_prediction_2.csv', index=False)


print('coef of IVs',pd.DataFrame({"columns":list(df_train.drop(['Risk'], axis=1).columns)[1:], "coef":list(clf.coef_.T)}))

data_evaluation = pd.read_csv('/Users/Wenqing Shi/Desktop/CIS5570project/logistic_regression_prediction_2.csv')



# 准确率
from sklearn.metrics import accuracy_score
y_pred = data_evaluation["Risk"]
y_true = data_test["Risk"]

print('accuracy is',accuracy_score(y_true, y_pred))

from sklearn import metrics

print('precision for micro is', metrics.precision_score(y_true, y_pred, average='micro'))  # 微平均，精确率


print('precision for macro is',metrics.precision_score(y_true, y_pred, average='macro'))  # 宏平均，精确率

#recall rate

print('recall rate  for micro is',metrics.recall_score(y_true, y_pred, average='micro'))

print('recall rate  for macro is',metrics.recall_score(y_true, y_pred, average='macro'))


# 分类报告：precision/recall/fi-score/均值/分类个数
from sklearn.metrics import classification_report
```

```
target_names = ['class 0', 'class 1']
print('evaluation summary is \n', classification_report(y_true, y_pred,
target_names=target_names))

#kappa score
from sklearn.metrics import cohen_kappa_score

print('kappa score is',cohen_kappa_score(y_true, y_pred))


from sklearn.metrics import roc_curve, auc
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(2):
    fpr[i], tpr[i], _ = roc_curve(y_true, y_pred)
    roc_auc[i] = auc(fpr[i], tpr[i])

# Generate ROC curve values: fpr, tpr, thresholds

plt.figure()
# Plot ROC curve
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr[1], tpr[1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```