```
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())

sns.set()
cols = ['Age', 'Sex','Job', 'Housing', 'Saving_accounts', 'Checking_accounts', 'Credit_amount',
'Duration', 'Purpose', 'Risk']
sns.pairplot(data_train[cols],size=2.5)
plt.show()

data_train.Risk.value_counts().plot(kind='bar')
plt.title("Count of good and bad")
plt.ylabel("Count")
plt.ylabel("Risk")
plt.grid(b=True, which='major', axis='both')


Risk_1 = data_train.Age[data_train.Risk == 1].value_counts()
df=pd.DataFrame({'good':Risk_1})
df.plot(kind='bar', stacked=True)
plt.title("Risk good count of age")
plt.xlabel("Age")
plt.ylabel("count of good")
plt.grid(b=False, which='major', axis='both')


Risk_0 = data_train.Age[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'bad':Risk_0})
df.plot(kind='bar', stacked=True)
plt.title("Risk bad count of age")
plt.xlabel("Age")
plt.ylabel("count of bad")
plt.grid(b=False, which='major', axis='both')
```

```
Risk_amount1 = data_train.Credit_amount[data_train.Risk == 1].value_counts()
df=pd.DataFrame({'good':Risk_1})
df.plot(kind='bar', stacked=True)
plt.title("Risk good count of credit amount")
plt.xlabel("Credit_amount")
plt.ylabel("count of good")
plt.grid(b=False, which='major', axis='both')


Risk_amount0 = data_train.Credit_amount[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'bad':Risk_0})
df.plot(kind='bar', stacked=True)
plt.title("Risk bad count of credit amount")
plt.xlabel("Credit_amount")
plt.ylabel("count of bad")
plt.grid(b=False, which='major', axis='both')


Risk_month1 = data_train.Duration[data_train.Risk == 1].value_counts()
df=pd.DataFrame({'good':Risk_1})
df.plot(kind='bar', stacked=True)
plt.title("Risk good count of duration")
plt.xlabel("Duration")
plt.ylabel("count of good")
plt.grid(b=False, which='major', axis='both')


Risk_month0 = data_train.Duration[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'bad':Risk_0})
df.plot(kind='bar', stacked=True)
plt.title("Risk bad count of duration")
plt.xlabel("Duration")
plt.ylabel("count of bad")
plt.grid(b=False, which='major', axis='both')


plt.subplot2grid((2,2),(0,0))
data_train.Job.value_counts().plot(kind='bar')
plt.title("Count of Job")
plt.ylabel("Count",fontproperties = 'SimHei')
plt.grid(b=False, which='major', axis='both')
```

```
plt.subplot2grid((2,2),(0,1))
data_train.Housing.value_counts().plot(kind='bar')
plt.title("Count of Housing")
plt.ylabel("Count",fontproperties = 'SimHei')
plt.grid(b=False, which='major', axis='both')

plt.subplot2grid((2,2),(1,0))
data_train.Saving_accounts.value_counts().plot(kind='bar')
plt.title("Count of Housing")
plt.ylabel("Count",fontproperties = 'SimHei')
plt.grid(b=False, which='major', axis='both')

plt.subplot2grid((2,2),(1,1))
data_train.Purpose.value_counts().plot(kind='bar')
plt.title("Count of Purpose")
plt.ylabel("Count",fontproperties = 'SimHei')
plt.grid(b=False, which='major', axis='both')


Risk_sex1 = data_train.Sex[data_train.Risk == 1].value_counts()
Risk_sex0 = data_train.Sex[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'good':Risk_sex1, 'bad':Risk_sex0})
df.plot(kind='bar', stacked=True)
plt.title("Risk of Sex")
plt.ylabel("Count")
plt.xlabel("Sex")

Risk_job1 = data_train.Job[data_train.Risk == 1].value_counts()
Risk_job0 = data_train.Job[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'good':Risk_job1, 'bad':Risk_job0})
df.plot(kind='bar', stacked=True)
plt.title("Risk of job")
plt.xlabel("Job")
plt.ylabel("Count")

Risk_house1 = data_train.Housing[data_train.Risk == 1].value_counts()
Risk_house0 = data_train.Housing[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'good':Risk_house1, 'bad':Risk_house0})
df.plot(kind='bar', stacked=True)
plt.title("Risk of housing")
plt.xlabel("Housing")
plt.ylabel("Count")

Risk_save1 = data_train.Saving_accounts[data_train.Risk == 1].value_counts()
```

```
Risk_save0 = data_train.Saving_accounts[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'good':Risk_save1, 'bad':Risk_save0})
df.plot(kind='bar', stacked=True)
plt.title("Risk of saving accounts")
plt.xlabel("Saving_accounts")
plt.ylabel("Count")

Risk_check1 = data_train.Checking_accounts[data_train.Risk == 1].value_counts()
Risk_check0 = data_train.Checking_accounts[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'good':Risk_check1, 'bad':Risk_check0})
df.plot(kind='bar', stacked=True)
plt.title("Risk of checking accounts")
plt.xlabel("Checking_accounts")
plt.ylabel("Count")

Risk_purpose1 = data_train.Purpose[data_train.Risk == 1].value_counts()
Risk_purpose0 = data_train.Purpose[data_train.Risk == 0].value_counts()
df=pd.DataFrame({'good':Risk_purpose1, 'bad':Risk_purpose0})
df.plot(kind='bar', stacked=True)
plt.title("Risk of purpose")
plt.xlabel("Purpose")
plt.ylabel("Count")

plt.show()



# print(data_train["Saving_accounts"].unique())
data_train["Saving_accounts"] = data_train["Saving_accounts"].fillna('no_info')

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

# 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
sex_mapping = {'female':1, 'male':2}
data_train['Sex'] = data_train['Sex'].map(sex_mapping)

saving_mapping = {'no info':0,'little':1, 'moderate':2, 'rich':3, 'quiet rich':4}
data_train['Saving_accounts'] = data_train['Saving_accounts'].map(saving_mapping)
```

```
checking_mapping = {'no info':0,'little':1, 'moderate':2, 'rich':3}
data_train['Checking_accounts'] = data_train['Checking_accounts'].map(checking_mapping)

purpose_mapping = {'radio/TV':0,'furniture/equipment':1, 'vacation/others':2, 'repirs':3,'car':4,
'domestic appliances':5,'education':6, 'business':7}
data_train['Purpose'] = data_train['Purpose'].map(purpose_mapping)



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

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')
Risk = pd.get_dummies(data_train.Risk, drop_first=False, prefix='Risk')
df = pd.concat(
   [data_train, dummy_Sex, dummy_Job, dummy_Housing, dummy_Saving, dummy_Checking,
dummy_Purpose], axis=1)
df.drop(['Job', 'Sex', 'Housing', 'Saving_accounts', 'Checking_accounts', 'Purpose'], axis=1,
      inplace=True)
# Excluding the missing columns


import sklearn.preprocessing as preprocessing

scaler = preprocessing.StandardScaler()

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

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_.*|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

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

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

# Purpose to Dummies Variable
sex_mapping = {'female':1, 'male':2}
data_test['Sex'] = data_test['Sex'].map(sex_mapping)

saving_mapping = {'no info':0,'little':1, 'moderate':2, 'rich':3, 'quite rich':4}
data_test['Saving_accounts'] = data_test['Saving_accounts'].map(saving_mapping)
```

```
checking_mapping = {'no info':0,'little':1, 'moderate':2, 'rich':3}
data_test['Checking_accounts'] = data_test['Checking_accounts'].map(checking_mapping)

purpose_mapping = {'radio/TV':0,'furniture/equipment':1, 'vacation/others':2, 'repirs':3,'car':4,
'domestic appliances':5,'education':6, 'business':7}
data_test['Purpose'] = data_test['Purpose'].map(purpose_mapping)

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_Sex, dummy_Job, dummy_Housing, dummy_Saving, dummy_Checking,
dummy_Purpose], axis=1)
df_test.drop(['Job', 'Sex', 'Housing', 'Saving_accounts', 'Checking_accounts',
'Purpose','Risk','ID'], axis=1,
          inplace=True)
# Excluding the missing columns

import sklearn.preprocessing as preprocessing


df_test = pd.concat([df_test], axis=1)

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

test = df_test.filter(

regex='Age|Sex_.*|Job_.*|Housing_.*|Saving_accounts_.*|Checking_accounts_.*|Credit_amo
unt|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_1.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_prediction = pd.read_csv('/Users/Wenqing
Shi/Desktop/CIS5570project/logistic_regression_prediction_1.csv')
```

```
# 准确率
from sklearn.metrics import accuracy_score
y_pred = data_prediction["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(0,2):
    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()
```