```
# CONSTRUCT A DATAFRAM
"""

import pandas as pd
import numpy as np
credit_df = pd.read_csv('German Credit Data.csv' )
credit_df.info()

credit_df.head()

credit_df.status.value_counts()

X_features = list( credit_df.columns )
X_features.remove( 'status' )
X_features

encoded_credit_df = pd.get_dummies(credit_df[X_features],drop_first = True)

list(encoded_credit_df.columns)

encoded_credit_df[['checkin_acc_A12','checkin_acc_A13','checkin_acc_A14']].head(5)

import statsmodels.api as sm
Y = credit_df.status
X = sm.add_constant(encoded_credit_df)

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size = 0.3,random_state = 42)

logit = sm.Logit(y_train, X_train)
logit_model = logit.fit()

logit_model.summary2()

def get_significant_vars( lm ):
    #Store the p-values and corresponding column names in a dataframe
    var_p_vals_df = pd.DataFrame( lm.pvalues )
    var_p_vals_df['vars'] = var_p_vals_df.index
    var_p_vals_df.columns = ['pvals', 'vars']
    # Filter the column names where p-value is less than 0.05
    return list( var_p_vals_df[var_p_vals_df.pvals <= 0.05]['vars'] )

significant_vars = get_significant_vars( logit_model )
significant_vars

final_logit = sm.Logit( y_train,
sm.add_constant( X_train [significant_vars] ) ).fit()

final_logit.summary2()

y_pred_df=pd.DataFrame( {"actual": y_test,"predicted_prob":
final_logit.predict(sm.add_constant( X_test[significant_vars]))})

y_pred_df.sample(10, random_state = 42)
```

```
y_pred_df['predicted'] = y_pred_df.predicted_prob.map(lambda x: 1 if x > 0.5 else 0)
y_pred_df.sample(10, random_state = 42)

# Commented out IPython magic to ensure Python compatibility.
import matplotlib.pyplot as plt
import seaborn as sn
# %matplotlib inline

from sklearn import metrics
def draw_cm( actual, predicted ):
    ## Cret
    cm = metrics.confusion_matrix( actual, predicted, [1,0] )
    sn.heatmap(cm, annot=True, fmt='.2f',xticklabels = ['Bad credit', 'Good Credit'],yticklabels =
["Bad credit", "Good Credit"] )
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

draw_cm( y_pred_df.actual,y_pred_df.predicted )

print( metrics.classification_report( y_pred_df.actual,y_pred_df.predicted ) )
```