credit-risk-analysis

May 30, 2024

Credit Risk Analysis for extending Bank Loans

Getting Started with Credit Risk Prediction

Import Libraries

[3]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler  # Import the StandardScaler␣
 ↪class
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score

%matplotlib inline
import os
```

Working with Data

[5]:
```
df = pd.read_csv('/content/bankloans.csv')
df.head()
```

[5]:
```
   age  ed  employ  address  income  debtinc   creddebt   othdebt  default
0   41   3      17       12     176      9.3  11.359392  5.008608      1.0
1   27   1      10        6      31     17.3   1.362202  4.000798      0.0
2   40   1      15       14      55      5.5   0.856075  2.168925      0.0
3   41   1      15       14     120      2.9   2.658720  0.821280      0.0
4   24   2       2        0      28     17.3   1.787436  3.056564      1.0
```

[6]:
```
df.isnull().sum()
```

[6]:
```
age           0
ed            0
employ        0
address       0
income        0
debtinc       0
creddebt      0
othdebt       0
default     450
dtype: int64
```

[7]:
```
df.value_counts()
```

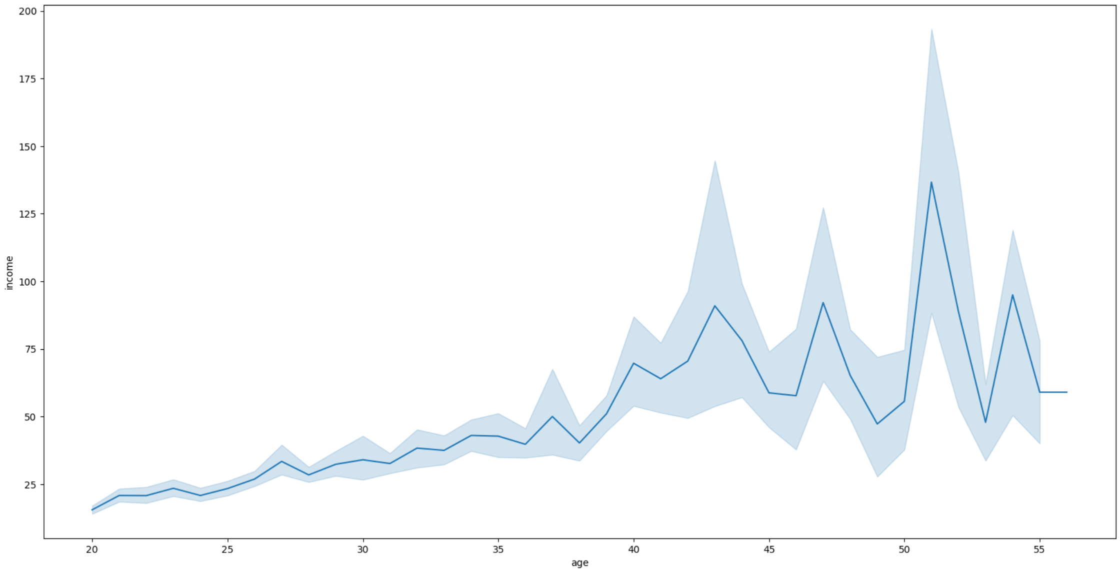
[7]:
```
age  ed  employ  address  income  debtinc  creddebt  othdebt    default
20   1   4       0        14      9.7      0.200984  1.157016   1.0        1
39   1   10      4        31      4.8      0.184512  1.303488   0.0        1
         0       8        39      7.9      1.066026  2.014974   0.0        1
         2       15       22      23.1     1.915914  3.166086   1.0        1
         4       9        38      6.5      1.178190  1.291810   0.0        1
                                                                          ..
30   2   8       4        56      6.4      0.333312  3.250688   0.0        1
         10      4        22      16.1     1.409716  2.132284   0.0        1
         12      9        68      20.1     2.856612  10.811388  0.0        1
                          98      7.2      2.935296  4.120704   0.0        1
56   1   11      20       59      15.0     4.672800  4.177200   0.0        1
Name: count, Length: 700, dtype: int64
```

[8]:
```
df = df.dropna()
```

Visualise Data

[9]:
```
fig,ax=plt.subplots(figsize=(20,10))
sns.lineplot(x="age", y ="income", data = df, ax=ax)
```

[9]:
```
<Axes: xlabel='age', ylabel='income'>
```

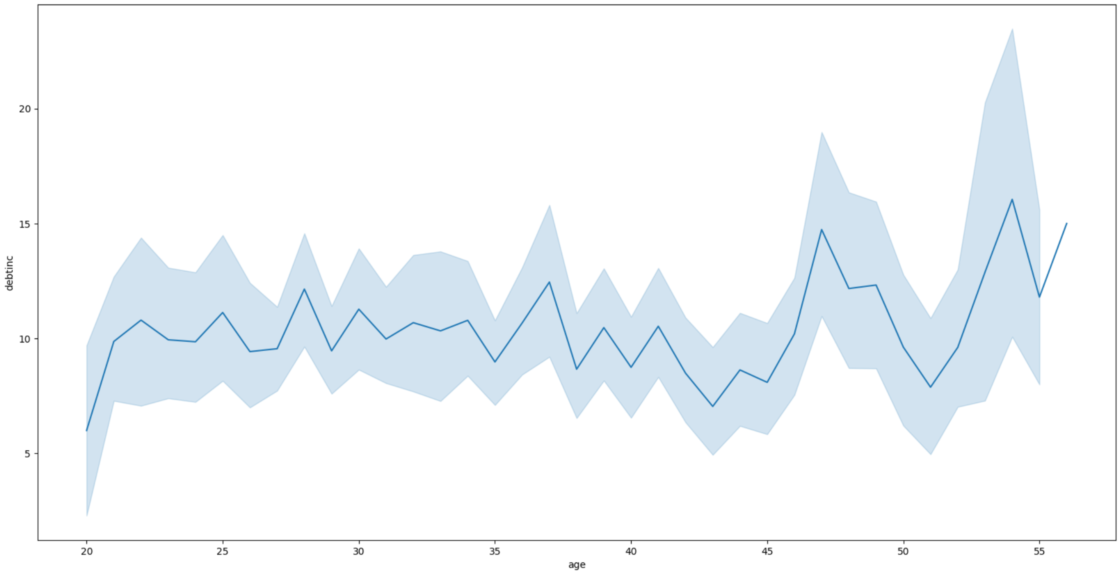


[10]:

```
fig,ax=plt.subplots(figsize=(20,10))
sns.lineplot(x="age", y ="debtinc", data = df, ax=ax)
```

[10]:

```
<Axes: xlabel='age', ylabel='debtinc'>
```



[11]:

```
df['default'].value_counts()
```

[11]:

```
default
0.0    517
```

```
1.0    183
Name: count, dtype: int64
```

Train Test Split

[12]:
```
x=df.drop(['default'], axis=1)
y=df['default']
```

[13]:
```
xtrain,xtest,ytrain,ytest =train_test_split(x, y, test_size=0.2, random_state=42)
```

[14]:
```
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest= sc.fit_transform(xtest)
```

Creating Model

Random Forest

[15]:
```
rfc = RandomForestClassifier(n_estimators=200)
```

[16]:
```
rfc.fit(xtrain,ytrain)
```

[16]:
```
RandomForestClassifier(n_estimators=200)
```

[17]:
```
rfc.score(xtest,ytest)
```

[17]:
```
0.8
```

[18]:
```
rfc2 =cross_val_score(estimator=rfc, X=xtrain, y=ytrain, cv=10)
rfc2.mean()
```

[18]:
```
0.7821428571428573
```

SVM

[19]:
```
sv = SVC()
sv.fit(xtrain,ytrain)
```

[19]:
```
SVC()
```

[20]:
```
sv.score(xtest,ytest)
```

[20]:
```
0.7928571428571428
```

[21]:
```
model = GridSearchCV(sv, {
            'C': [0.1,0.2,0.4,0.8,1.2,1.8,4.0,7.0],
            'gamma': [0.1,0.4,0.8,1.0,2.0,3.0],
            'kernel': ['rbf', 'linear']
            }, scoring='accuracy', cv=10)
```

[22]:
```
model.fit(xtrain,ytrain)
```

[22]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': [0.1, 0.2, 0.4, 0.8, 1.2, 1.8, 4.0, 7.0],
                         'gamma': [0.1, 0.4, 0.8, 1.0, 2.0, 3.0],
                         'kernel': ['rbf', 'linear']},
             scoring='accuracy')
```

[23]:
```
model.best_params_
```

[23]:
```
{'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}
```

[24]:
```
model2 = SVC(C=0.1,gamma=0.1,kernel= 'linear' )
model2.fit(xtrain, ytrain)
model2.score(xtest, ytest)
```

[24]:
```
0.8214285714285714
```

Logistic Regression

[25]:
```
lr = LogisticRegression()
```

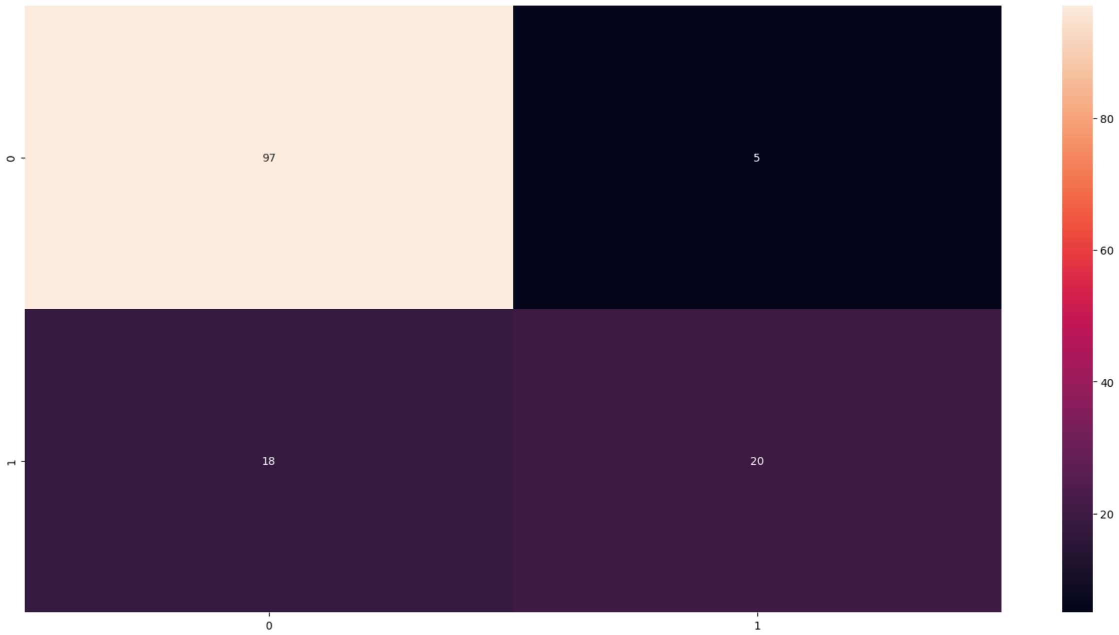
[29]:
```
# Fit the LogisticRegression model
lr.fit(xtrain, ytrain)

# Make predictions on the test data
yp = lr.predict(xtest)

# Create a confusion matrix
c = confusion_matrix(ytest, yp)

# Visualize the confusion matrix
fig, ax = plt.subplots(figsize=(20, 10))
sns.heatmap(c, ax=ax, annot = True)
```

[29]:
```
<Axes: >
```


97
5
18
20
0
1
0
1
80
60
40
20

```
[ ]:
```