PROJECT

INSTRUCTOR

RAJASEKHAR

TEAM MEMBERS

- Mr. Shantanu Jaiprakash Kamtalwar
- Mr. Kalangi.Pavan Venkat
- Mr. Sanket Deepak Taskar
- Ms. Sunita Basalingayya
- Ms. Kalyani Gedam
- Mr. Sumant Nag
- Mr. Vempali. Mani Ratnam

BANKRUPTCY PREDICTION

PROBLEM STATEMENT

OBJECTIVE:

This is a classification project, since the variable to predict is binary (bankruptcy or non-bankruptcy). The goal here is to model the probability that a business goes bankrupt from different features.

DATA DESCRIPTION

DATA FILE

The data file contains 7 features about 250 companies

The data set includes the following variables:

- 1. industrial_risk
- 2. management_risk
- 3. financial_flexibility
- 4. credibility
- 5. competitiveness
- 6. operating_risk

- Based on the above variables we are supposed to make prediction on target variable **class**
    - *class:*
        - *bankruptcy*
        - *non-bankruptcy*

ALL THE VARIABLES GIVE ARE CATEGORICAL IN NATURE

- The Following table displays all the categories in a given variable.

Variable Name	Low	Medium	High
industrial_risk	0	0.5	1
management_risk	0	0.5	1
financial flexibility	0	0.5	1
credibility	0	0.5	1
competitiveness	0	0.5	1
operating_risk	0	0.5	1

EDA

DATA DESCRIPTION

```
df.describe()
```

index	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk
count	250.0	250.0	250.0	250.0	250.0	250.0
mean	0.518	0.614	0.376	0.47	0.476	0.57
std	0.411	0.410	0.401	0.415	0.440	0.434
min	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.5	0.0	0.0	0.0	0.0
50%	0.5	0.5	0.5	0.5	0.5	0.5
75%	1.0	1.0	0.5	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0	1.0	1.0

This info is not much useful ass all the variables are ordinal in nature

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   industrial_risk         250 non-null    float64
 1    management_risk        250 non-null    float64
 2    financial_flexibility  250 non-null    float64
 3    credibility            250 non-null    float64
 4    competitiveness        250 non-null    float64
 5    operating_risk         250 non-null    float64
 6    class                  250 non-null    object 
dtypes: float64(6), object(1)
memory usage: 13.8+ KB
```

DUPLICATE DETECTION

`df.isnull().sum()`

```
industrial_risk            0
 management_risk           0
 financial_flexibility     0
 credibility               0
 competitiveness           0
 operating_risk            0
 class                     0
dtype: int64
```

which means no null values

DROPPING DUPLICATES

- `df_1 = df.drop_duplicates()` drops the duplicates
- `df_1.shape` gives us an out put of `(103,7)`

WHICH MEANS 147 DATAPOINTS ARE DUPLICATES IN THE GIVEN DATA.

Finally we are left out with 103 data points

AS ALL THE GIVEN DATA IS CATEGORICAL DATA **LETS HAVE A LOOK INTO THE FREQUENCIES.**

- Frequency distribution of categories in the variables.

index	0.0	0.5	1.0
industrial_risk	35	36	32
management_risk	29	32	42
financial_flexibility	33	36	34
credibility	22	42	39
competitiveness	24	26	53
operating_risk	38	24	41

frequency distribution of class(Target variable).

index	class
non-bankruptcy	78
bankruptcy	25

FREQUENCY DISTRIBUTION BETWEEN 'INDUSTRIAL_RISK', 'CLASS'

```
`df_1.value_counts(['industrial_risk', 'class'])`
```

```
industrial_risk  class
0.5              non-bankruptcy    30
0.0              non-bankruptcy    29
1.0              non-bankruptcy    19
                 bankruptcy        13
0.0              bankruptcy         6
0.5              bankruptcy         6
dtype: int64
```

FREQUENCY DISTRIBUTION BETWEEN 'MANAGEMENT_RISK', 'CLASS'

```
`df_1.value_counts(['management_risk', 'class'])`
```

```
management_risk  class
1.0              non-bankruptcy    28
0.0              non-bankruptcy    25
0.5              non-bankruptcy    25
1.0              bankruptcy        14
0.5              bankruptcy         7
0.0              bankruptcy         4
dtype: int64
```

FREQUENCY DISTRIBUTION BETWEEN 'FINANCIAL_FLEXIBILITY', 'CLASS'

```
`df_1.value_counts(['financial_flexibility', 'class'])`
```

```
financial_flexibility  class
0.5                    non-bankruptcy    35
1.0                    non-bankruptcy    33
0.0                    bankruptcy        23
                       non-bankruptcy    10
0.5                    bankruptcy         1
1.0                    bankruptcy         1
dtype: int64
```

FREQUENCY DISTRIBUTION BETWEEN 'CREDIBILITY', 'CLASS'

```
`df_1.value_counts(['credibility', 'class'])`
```

```
credibility  class
0.5          non-bankruptcy    37
1.0          non-bankruptcy    37
0.0          bankruptcy        18
0.5          bankruptcy         5
0.0          non-bankruptcy     4
1.0          bankruptcy         2
dtype: int64
```

FREQUENCY DISTRIBUTION BETWEEN 'COMPETITIVENESS', 'CLASS'

```
`df_1.value_counts(['competitiveness', 'class'])`
```

```
competitiveness  class
1.0              non-bankruptcy    53
0.5              non-bankruptcy    25
0.0              bankruptcy        24
0.5              bankruptcy         1
dtype: int64
```

FREQUENCY DISTRIBUTION BETWEEN 'OPERATING_RISK', 'CLASS'

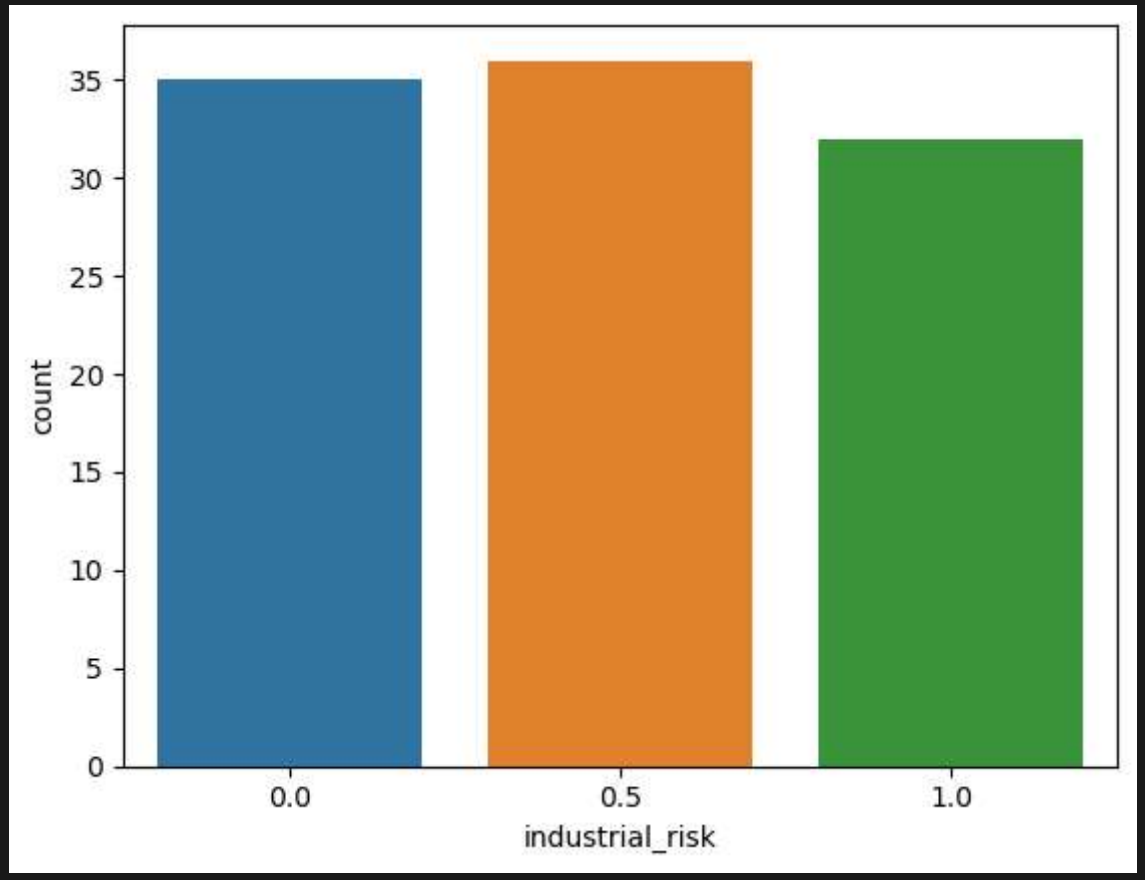
```
`df_1.value_counts(['operating_risk', 'class'])`
```

```
operating_risk  class
0.0             non-bankruptcy    31
1.0             non-bankruptcy    28
0.5             non-bankruptcy    19
1.0             bankruptcy        13
0.0             bankruptcy         7
0.5             bankruptcy         5
dtype: int64
```

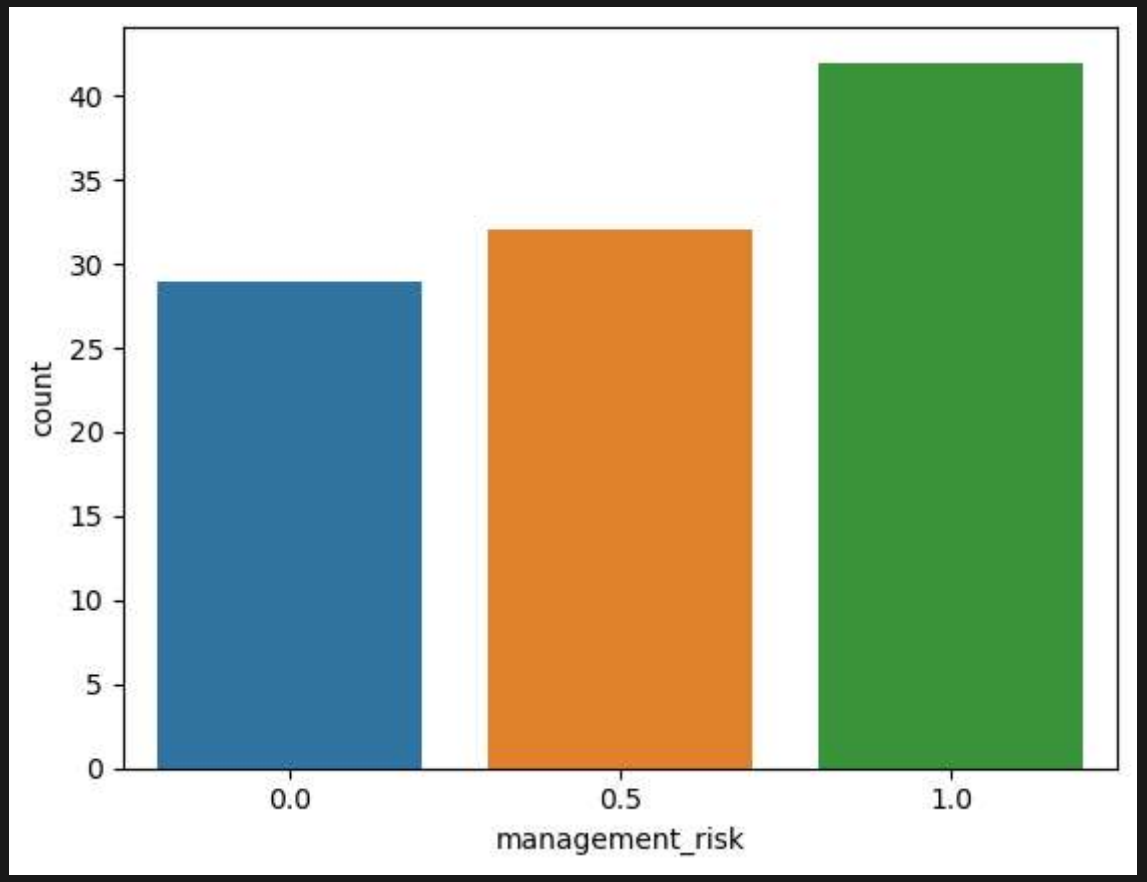
DATA VISUALIZATION

UNIVARIATE ANALYISIS

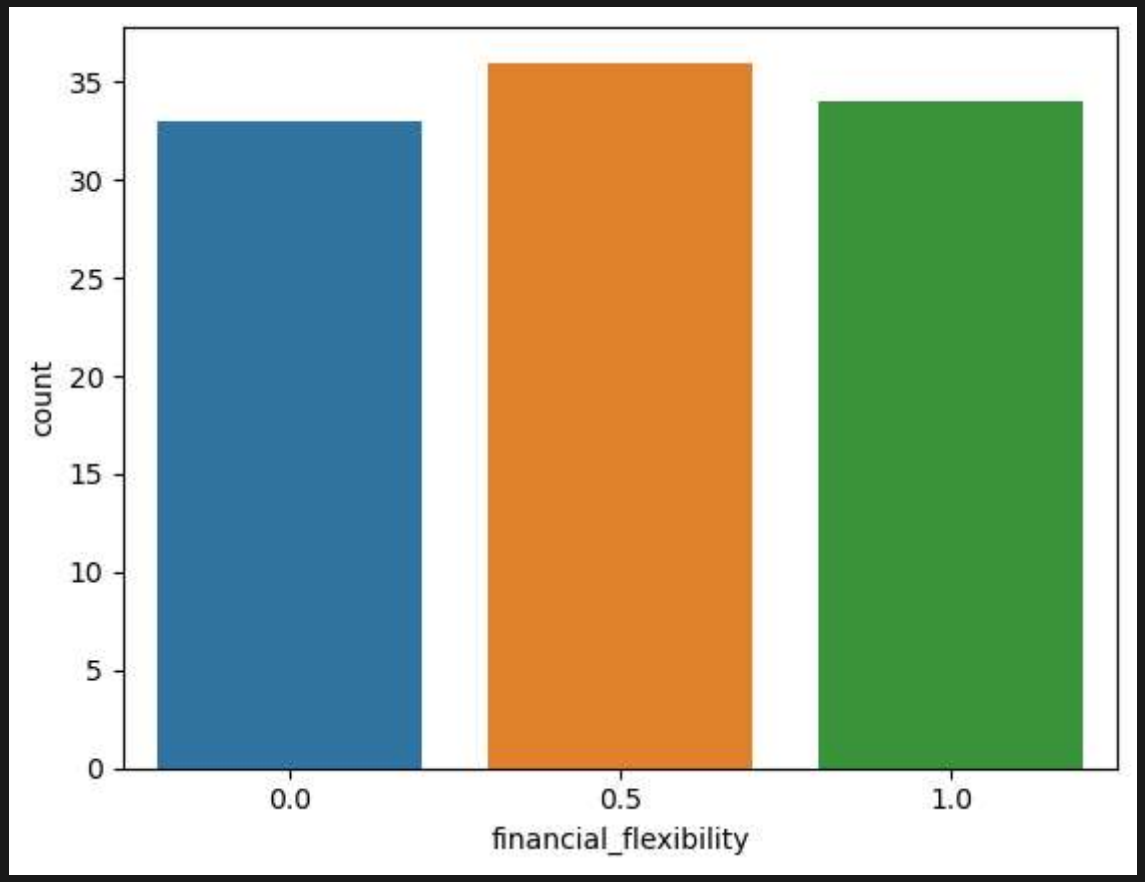
Count of each category in 'industrial_risk'



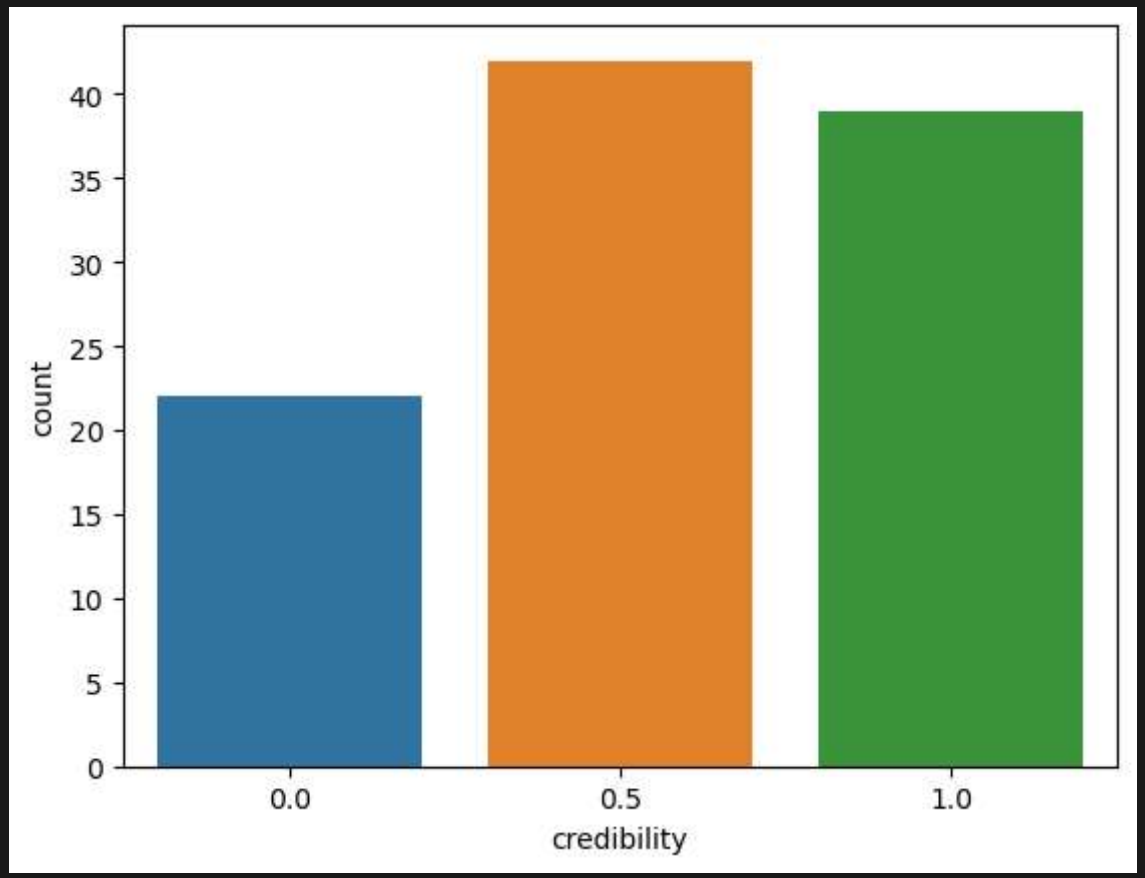
Count of each category in 'management_risk'



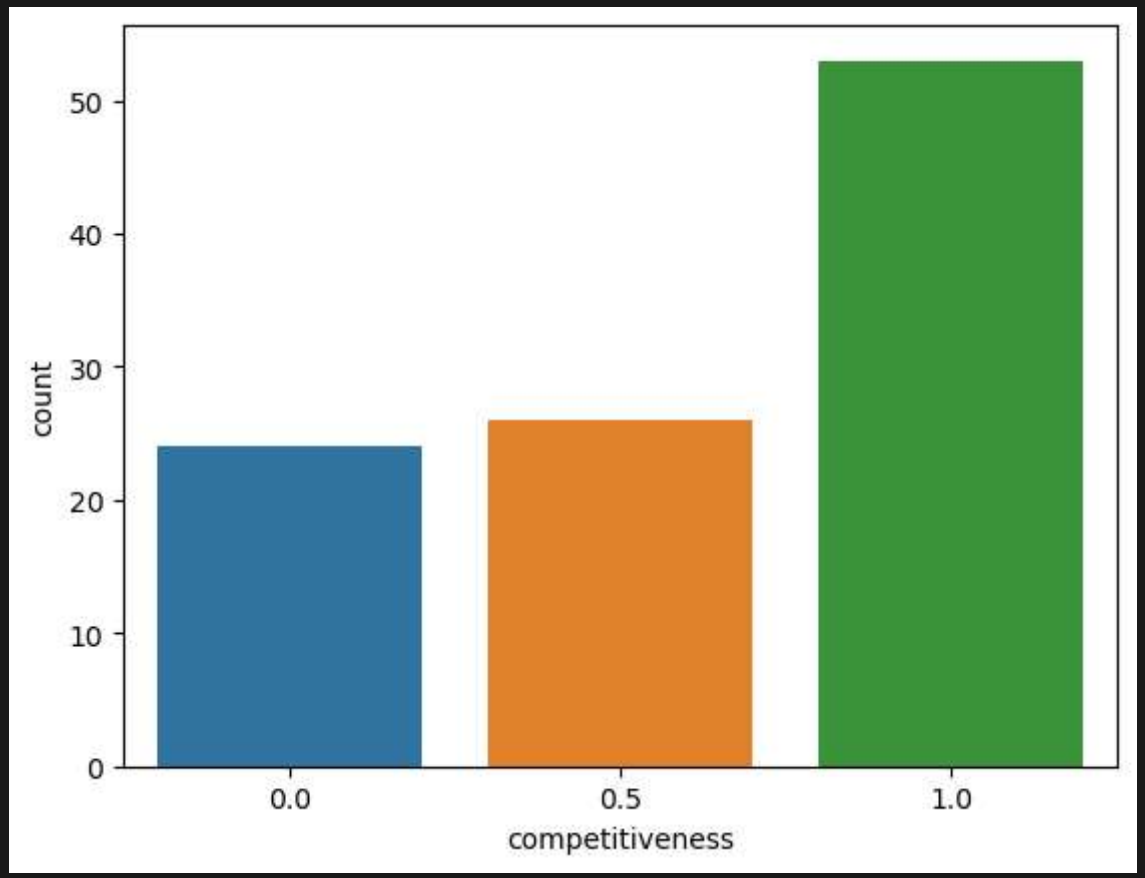
Count of each category in 'financial_flexibility'



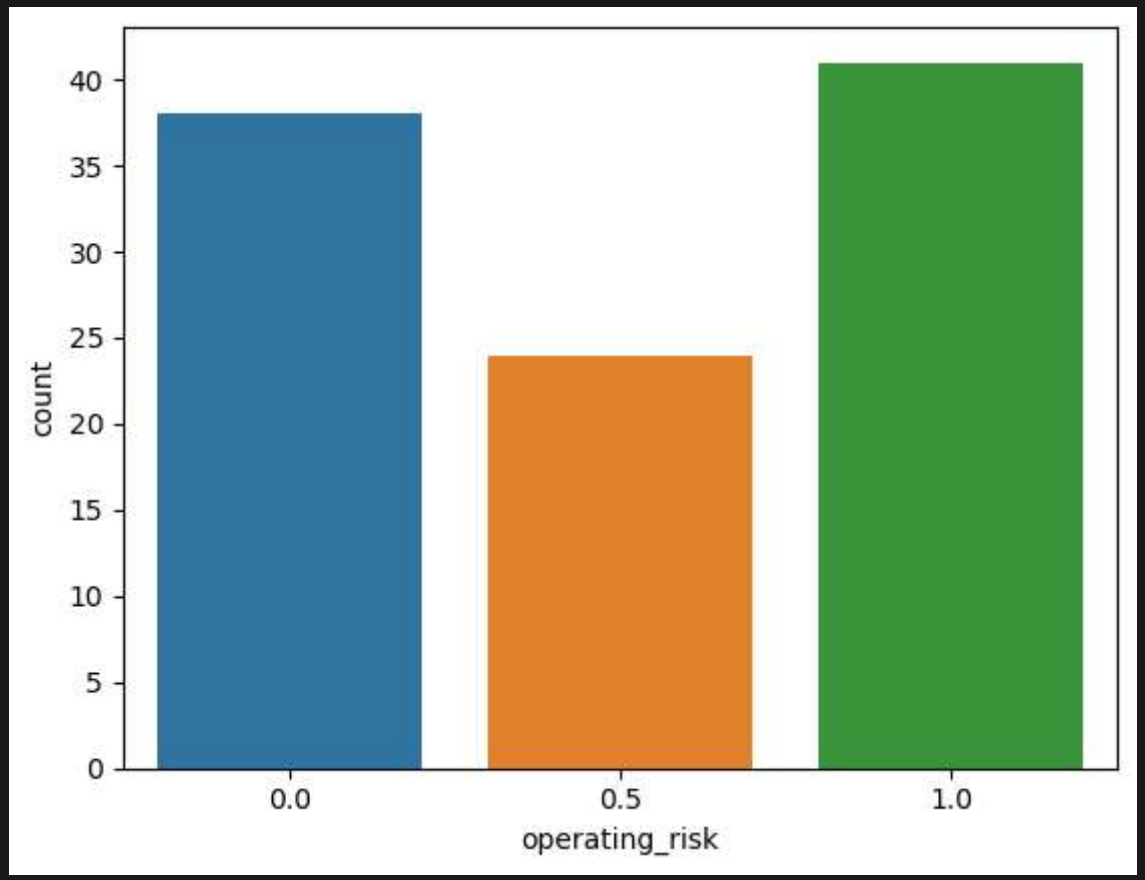
Count of each category in 'credibility'



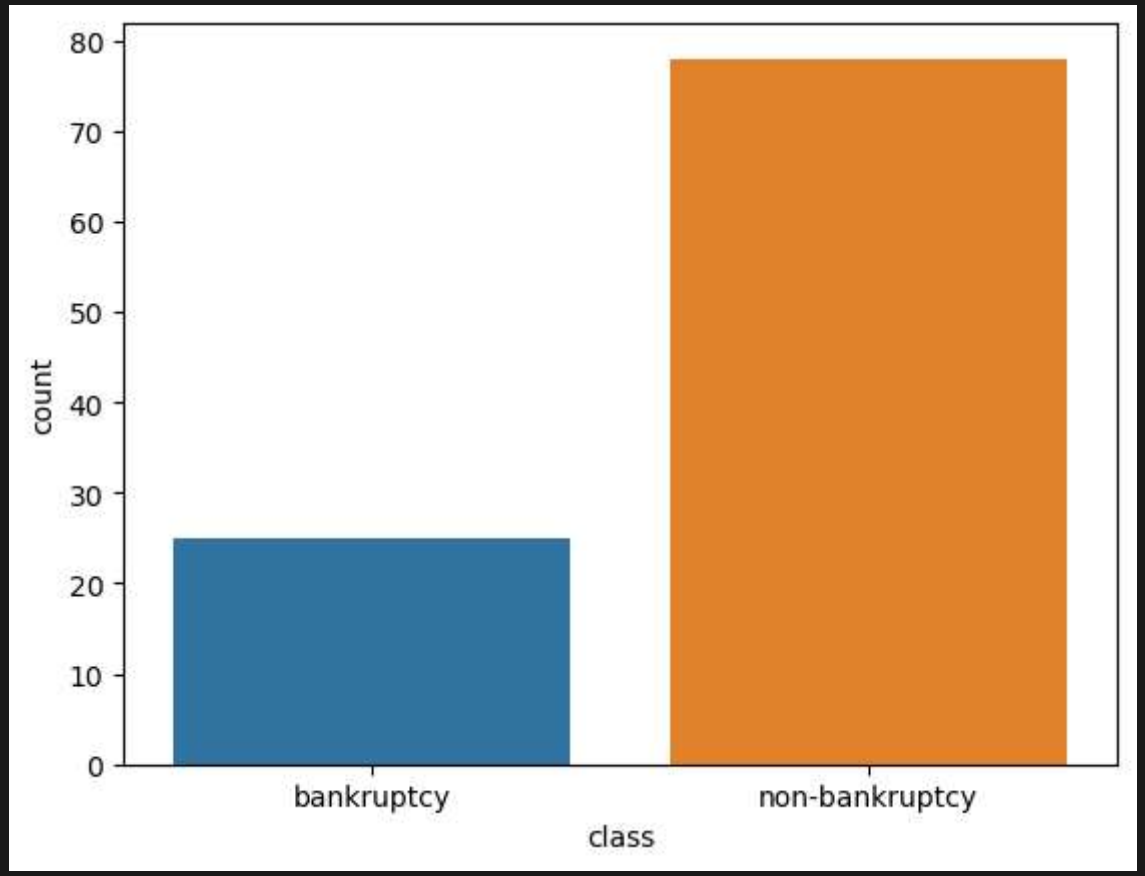
Count of each category in 'competitiveness'



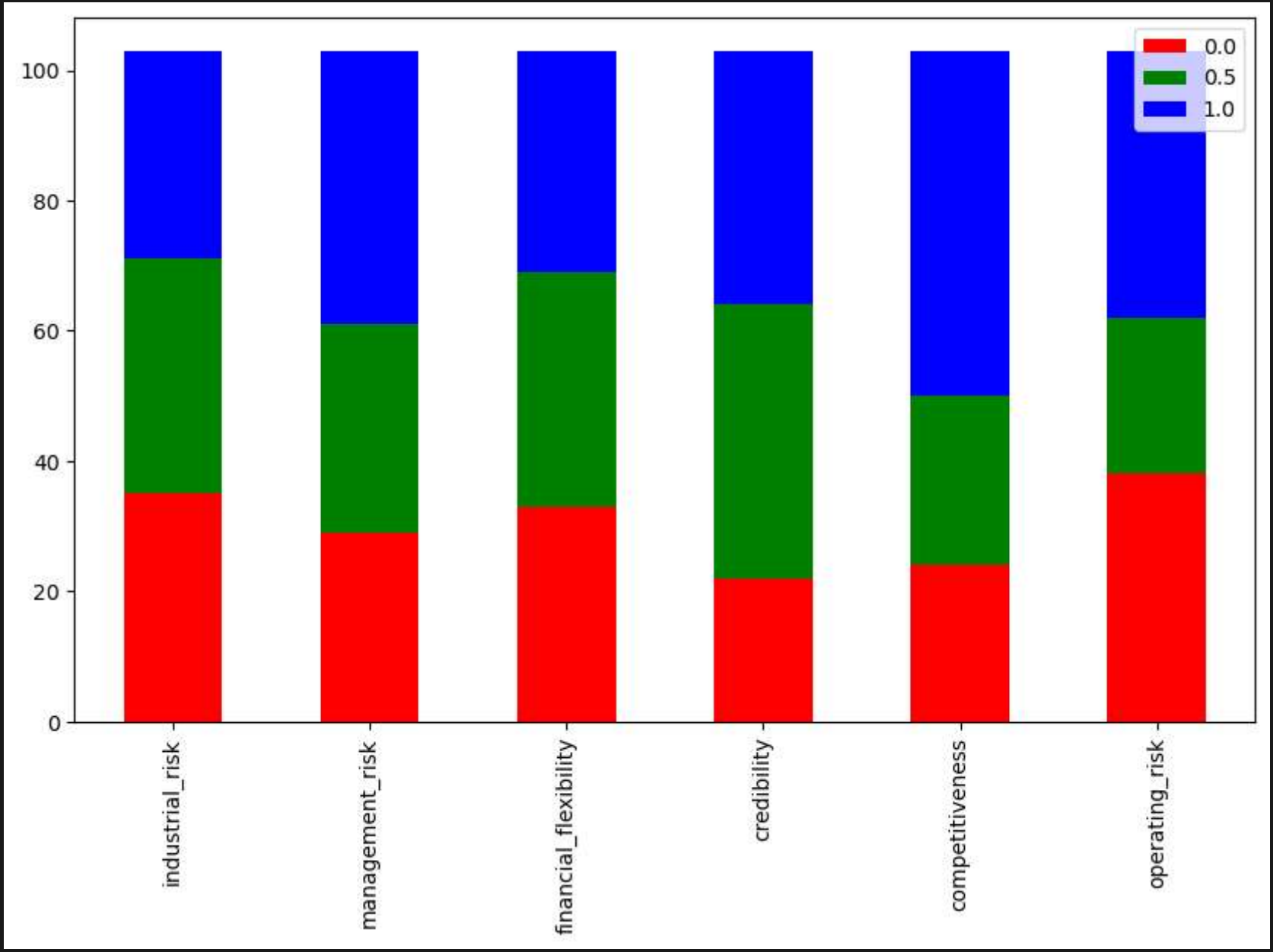
Count of each category in 'operating_risk'



Final count of target variable 'class'

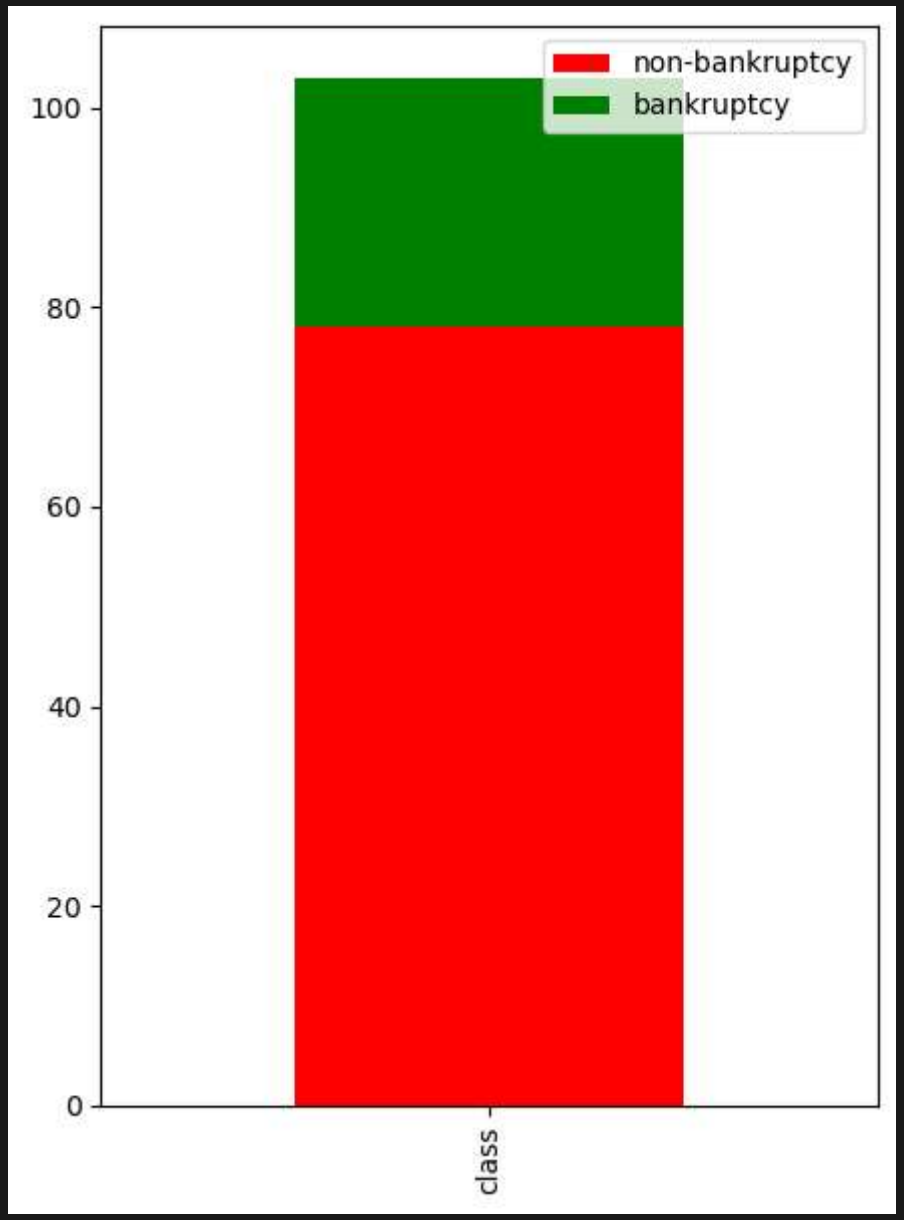


Here is the stacked graph of all the variable



All the variable have close to equal distribution and all the variables are important for making the decision.

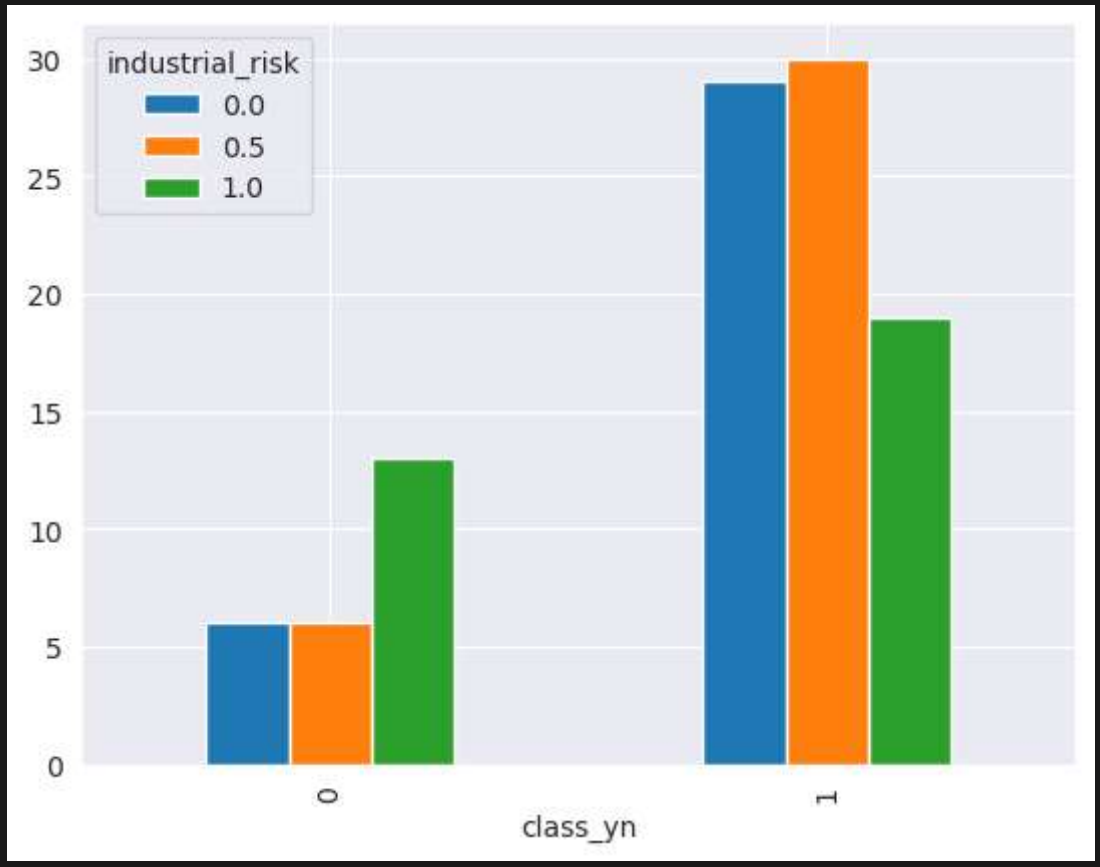
Stacked graph of target variable 'class'



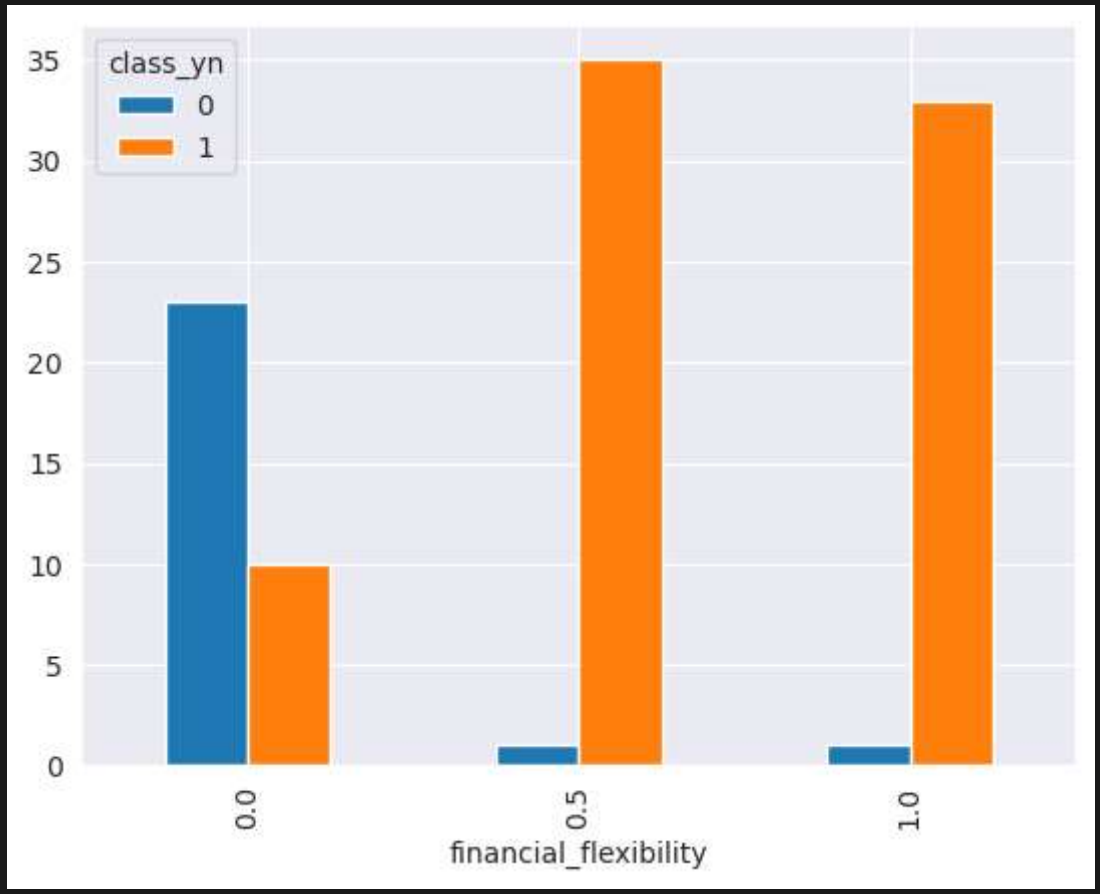
BIVARIATE ANALYSIS

GRAPHS BETWEEN 'CLASS' AND OTHER VARIABLES.

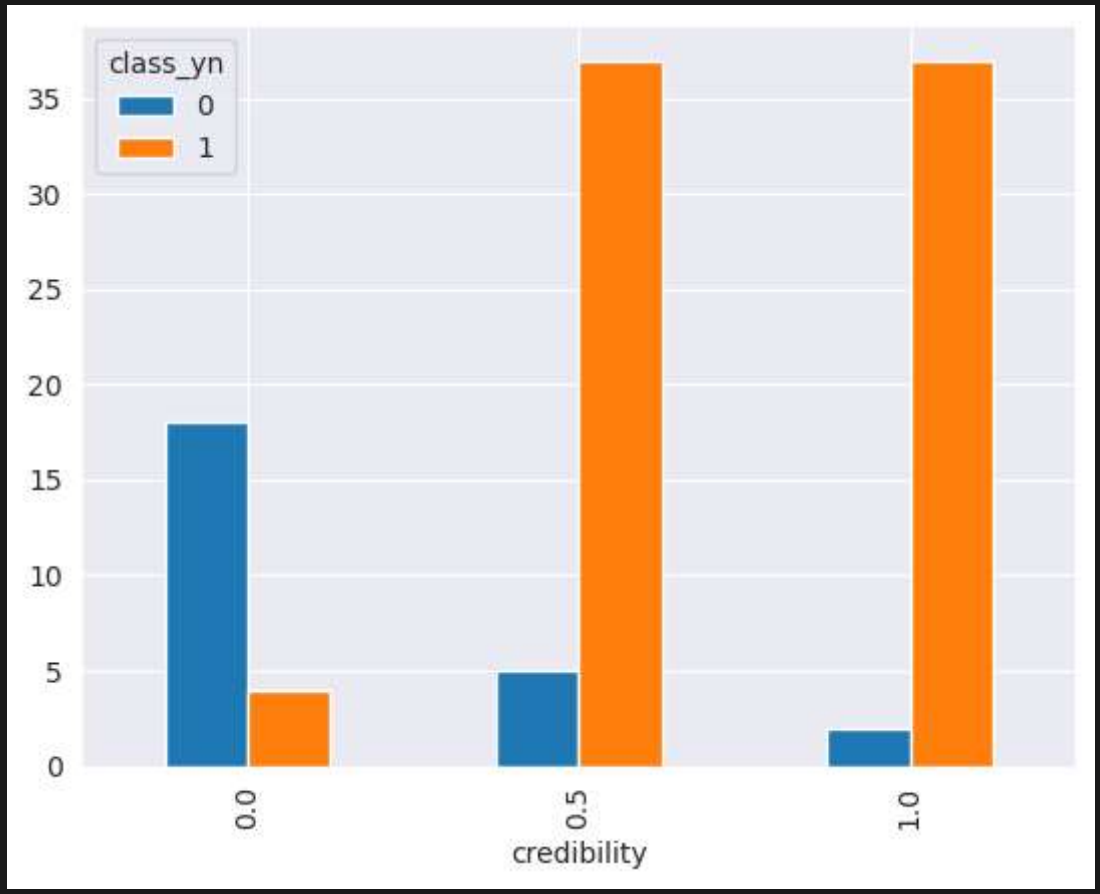
graph of 'Class' vs 'industrial_risk'



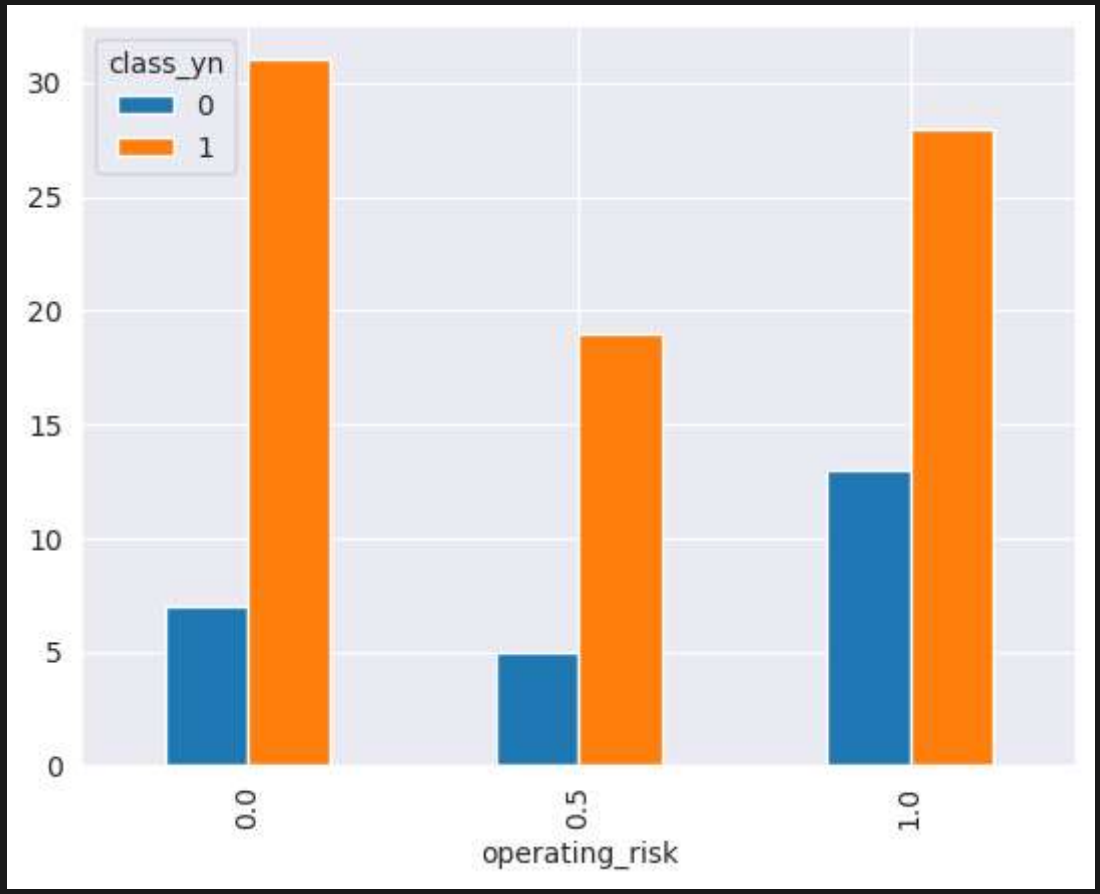
graph of 'Class' vs 'financial_flexibility'



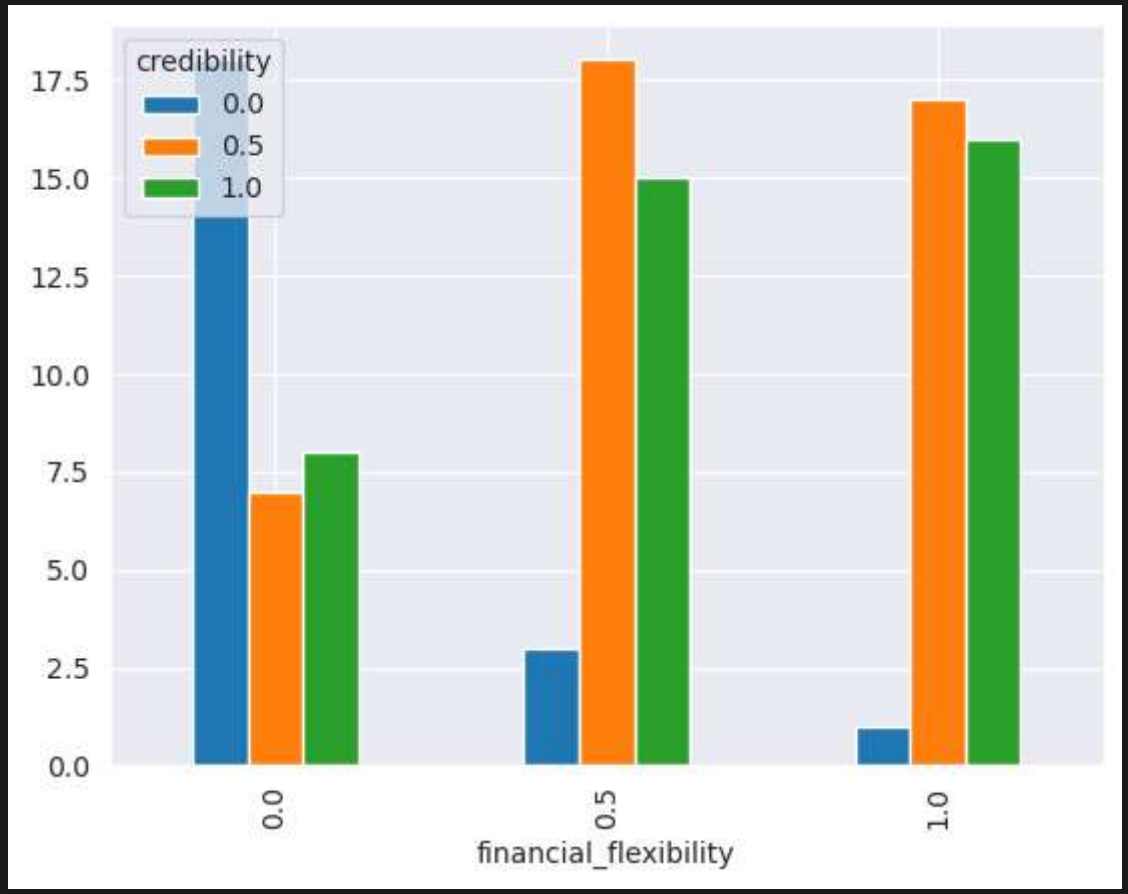
graph of 'Class' vs 'credibility'



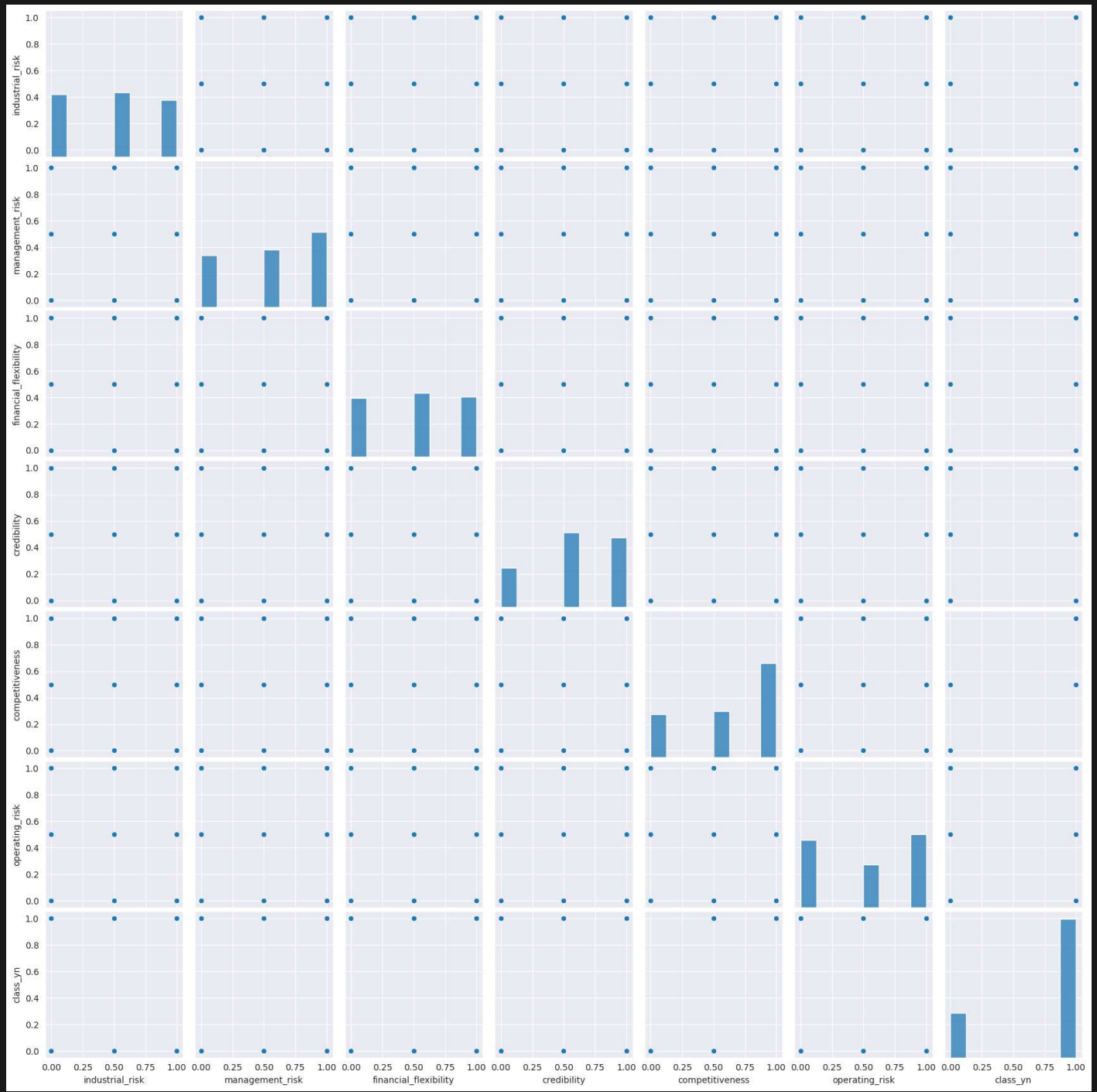
graph of 'Class' vs 'operation_risk'



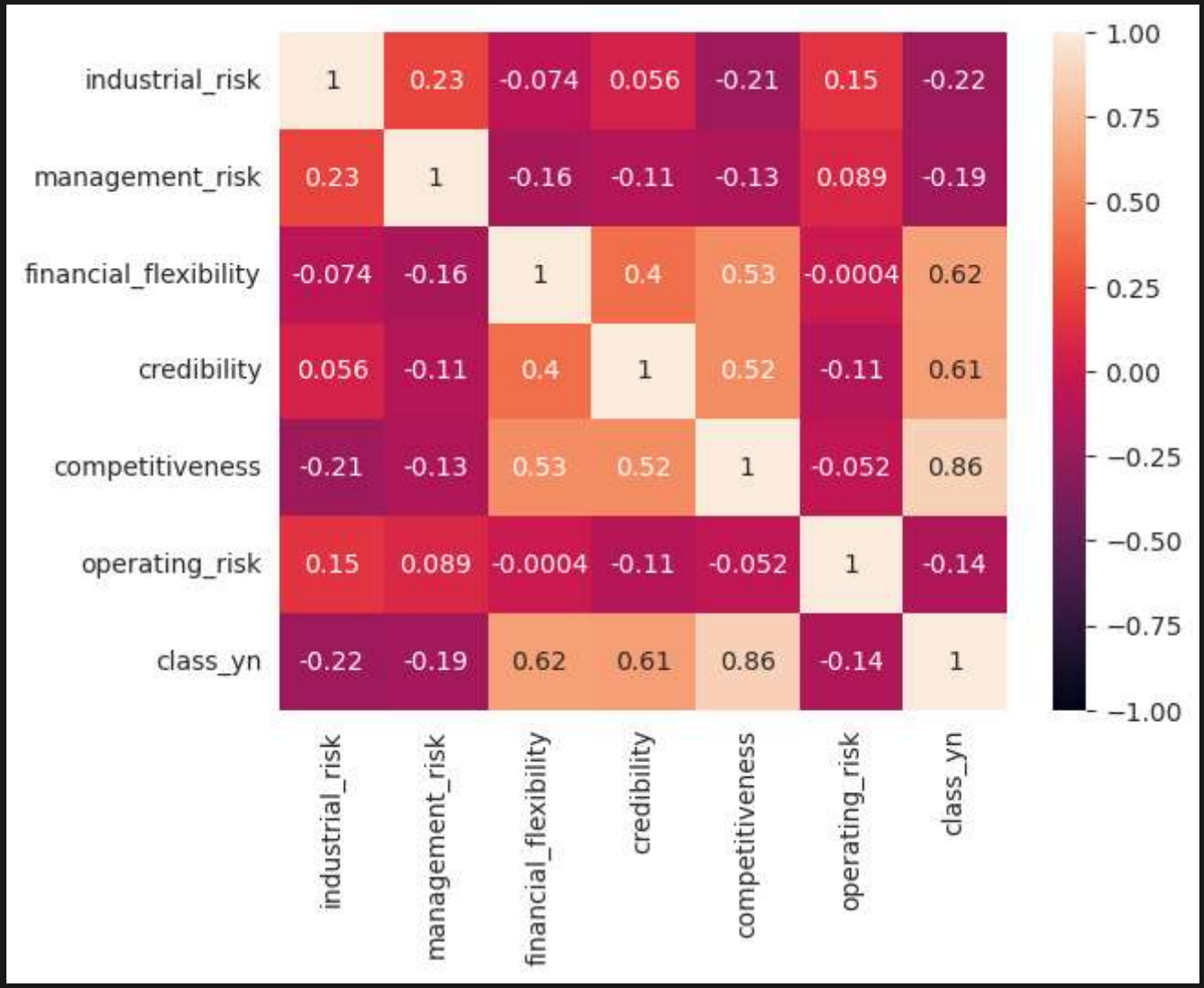
graph of 'credibility' vs 'financial_flexibility'



```
sns.set_style(style='darkgrid')
sns.pairplot(df_1_new)
```



Heat-map gives an insight into the feature engineering



As our data set is very small and we are taking all the features into our model

But if we need to pick top features 'financial_flexibility', 'credibility' and 'competitiveness' will be great features for modelling

MODEL BUILDING

BELOW MODELS ARE EXPLORED FOR THE GIVEN PROBLEM:

1. Logistic Regression
2. KNN
3. Naive Bayes Classifier
4. Support Vector Machine
5. DBSCAN
6. Decision Tree Classifier
7. Bagging Classifier
8. Random Forrest Classifier

SPLITTING THE DATA

The given data set is split into to two sets one as 'Training' and other as 'Test' set

- *Training Set $\rightarrow$ 75 % of total data*
- *Testing Set $\rightarrow$ 25% of total data*

LOGISTIC REGRESSION

Logistic Regression on training data .

```
from sklearn.linear_model import LogisticRegression
logisticclassifier = LogisticRegression()
logisticclassifier.fit(x_train, y_train)
logisticclassifier.coef_
```

Prediction on the trained model

```
y_pred = logisticclassifier.predict(x_test)
y_pred
```

Accuracy of the model

```
from sklearn.metrics import accuracy_score
logistic_acc = accuracy_score(y_test, y_pred)
logistic_acc
```

Accuracy = 0.9230769230769231

KNN MODEL

Building KNN Model

```
from sklearn.neighbors import KNeighborsClassifier as KNC
import math
math.sqrt(len(y_test))
KNN_classifier = KNC(n_neighbors =5, p = 2,
                     metric = 'euclidean')
KNN_classifier.fit(x_train, y_train)
```

Predicting with the test model

```
y_pred = KNN_classifier.predict(x_test)
y_pred
```

Evaluating the KNN Model

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import f1_score
print(f1_score(y_test, y_pred))
```

Accuracy of the KNN Model

```
from sklearn.metrics import accuracy_score
KNN_acc = accuracy_score(y_test, y_pred)
KNN_acc
```

Accuracy = 0.9615384615384616

NAIVE BAYES CLASSIFIER

Building Gaussian Naive Bayes classifier Model

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
GNB = GaussianNB()
MNB = MultinomialNB()
Naive_GNB = GNB.fit(x_train ,y_train)
y_pred = Naive_GNB.predict(x_test)
y_pred
```

Confusion Matrix and Accuracy of Gaussian Naive Bayes classifier Model

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
GNB_acc = accuracy_score(y_test , y_pred)
GNB_acc
```

Accuracy = 0.9615384615384616

Building Multinomial Naive Bayes classifier Model

```
Naive_MNB = MNB.fit(x_train ,y_train)
y_pred = Naive_MNB.predict(x_test)
y_pred
```

Confusion Matrix and Accuracy of Multinomial Naive Bayes classifier Model

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
from sklearn.metrics import accuracy_score
MNB_acc = accuracy_score(y_test , y_pred)
MNB_acc
```

Accuracy = 0.7692307692307693

SUPPORT VECTOR MACHINE

SVM linear Model

```
model_linear = SVC(kernel = 'linear')
model_linear.fit(x_train, y_train)
pred_test_linear = model_linear.predict(x_test)
np.mean(pred_test_linear==y_test)
```

Accuracy = 0.9615384615384616

SVM Poly Model

```
model_poly = SVC(kernel = "poly")
model_poly.fit(x_train,y_train)
pred_test_poly = model_poly.predict(x_test)
np.mean(pred_test_poly==y_test)
```

Accuracy = 0.9615384615384616

SVM Radial Basis Function Model

```
model_rbf = SVC(kernel = "rbf")
model_rbf.fit(x_train,y_train)
pred_test_rbf = model_rbf.predict(x_test)
np.mean(pred_test_rbf==y_test)
```

Accuracy = 0.9615384615384616

DBSCAN

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.8, min_samples=3)
dbscan.fit(X)
y=dbscan.labels_
y=pd.DataFrame(y)
y.value_counts()
```

DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth=(4))
dt.fit(x_train, y_train)
y_predtrain_dt = dt.predict(x_train)
y_predtest_dt = dt.predict(x_test)
print("Training Accuracy: ",accuracy_score(y_train,y_predtrain
print("Test Accuracy: ",accuracy_score(y_test,y_predtest_dt).r
```

- **Training Accuracy = 1.0**
- **Testing Accuracy = 1.0**

BAGGING CLASSIFIER

```
from sklearn.ensemble import BaggingClassifier
bag = BaggingClassifier(base_estimator=(dt),n_estimators=100,m
bag.fit(x_train, y_train)
y_predtrain_bag = bag.predict(x_train)
y_predtest_bag = bag.predict(x_test)
print("Training Accuracy: ",accuracy_score(y_train,y_predtrain
print("Test Accuracy: ",accuracy_score(y_test,y_predtest_bag).
```

- Training Accuracy = 1.0
- Testing Accuracy = 1.0

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(n_estimators=100,max_samples=0.9,
RFC.fit(x_train, y_train)

y_pred_train = RFC.predict(x_train)
y_pred_test = RFC.predict(x_test)

print("Training accuracy: ",accuracy_score(y_train,y_pred_trai
print("Test accuracy: ",accuracy_score(y_test,y_pred_test).rou
```

- Training Accuracy = 1.0
- Testing Accuracy = 1.0

ACCURACY OF DIFFERENT MODELS

Model	Accuracy
Logistic Regression	0.9902
KNN	0.9615
Naive Bayes (Gaussian)	0.9615
Naive Bayes (Multinomial)	0.7692
SVM (linear,poly, radial basis function)	0.9615
Decision Tree Classifier	1
Bagging Classifier	1
Random Forest Classifier	1

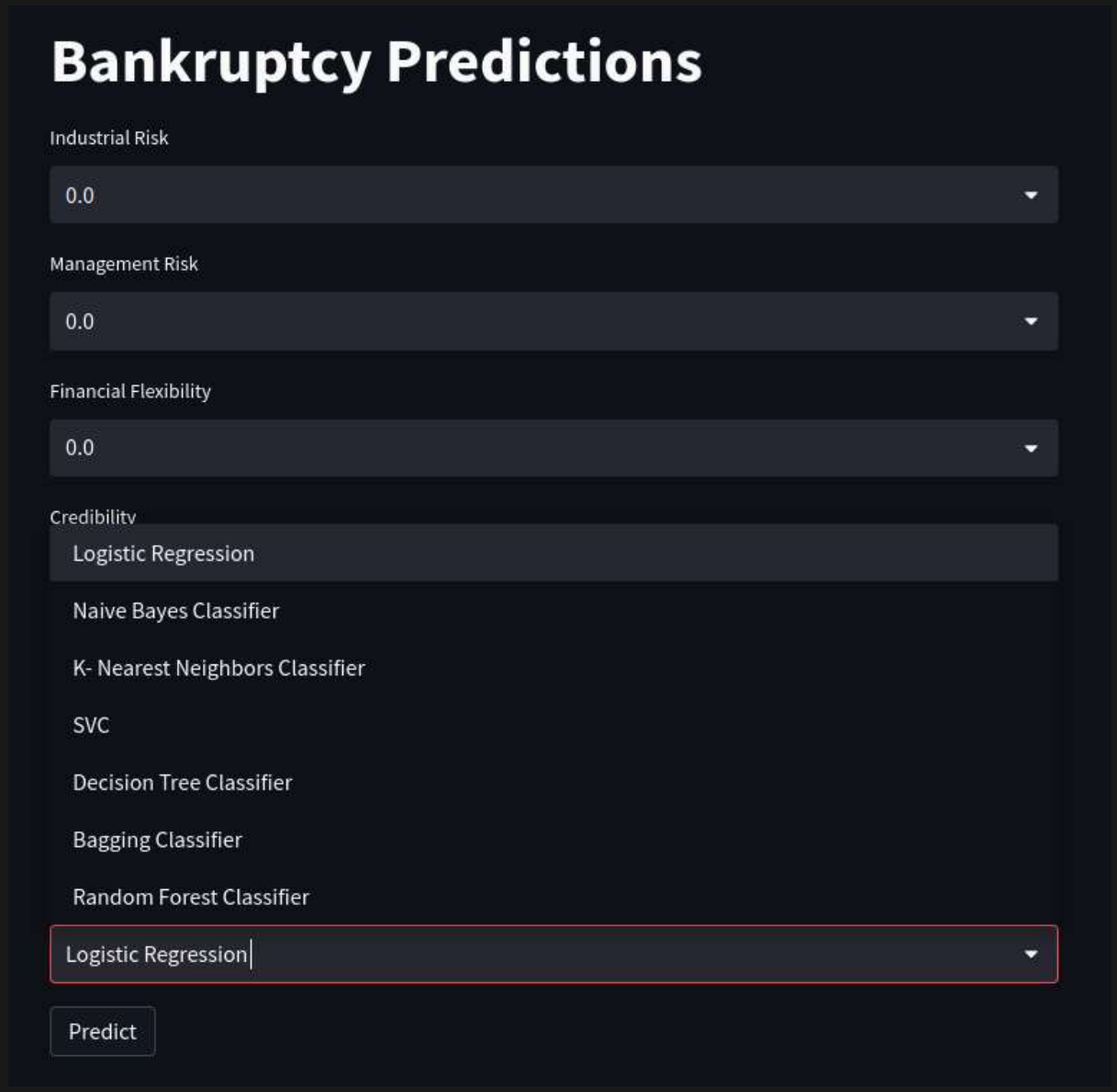
DEPLOYMENT

Using Pickle a binary of the Model is created

Deployment is done using streamlit and connecting it with binary.

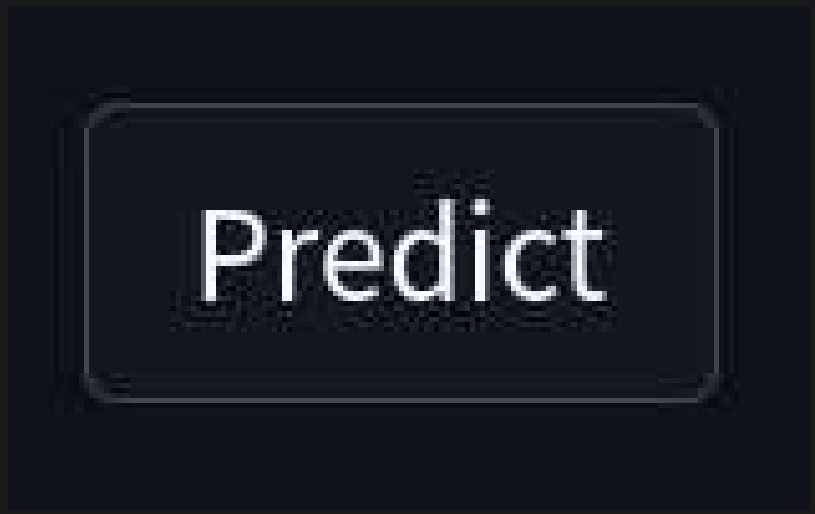
Final Model Page.

Select the model you are interested in.



Select the values of your choice for each attribute

CLICK ON PREDICT. FOR RESULTS.



Bankruptcy Predictions

Industrial Risk

0.0

Management Risk

0.0

Financial Flexibility

0.0

Credibility

0.0

Competitiveness

0.0

Operating Risk

0.0

Model

Logistic Regression

Predict

Predicted Result

Bankruptcy

CHALLENGES FACED

- Dealing with categorical variables
- Getting insight out of small data-set
- Binary conversion of model.
- Model deployment and UI design.

THANK YOU