Group member: Maisha Maliha , Nahid Anwar

Data Preperation and Exploration

Importing necessary python libraries

In [1]:
```
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.special import logit
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegressionCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, plot_
```

Date Load and Setup

In [2]:
```
df = pd.read_csv("SBAnational.csv", low_memory = False)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 899164 entries, 0 to 899163
Data columns (total 27 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   LoanNr_ChkDgt      899164 non-null  int64  
 1   Name               899150 non-null  object 
 2   City               899134 non-null  object 
 3   State              899150 non-null  object 
 4   Zip                899164 non-null  int64  
 5   Bank               897605 non-null  object 
 6   BankState          897598 non-null  object 
 7   NAICS              899164 non-null  int64  
 8   ApprovalDate       899164 non-null  object 
 9   ApprovalFY         899164 non-null  object 
 10  Term               899164 non-null  int64  
 11  NoEmp              899164 non-null  int64  
 12  NewExist           899028 non-null  float64
 13  CreateJob          899164 non-null  int64  
 14  RetainedJob        899164 non-null  int64  
 15  FranchiseCode      899164 non-null  int64  
 16  UrbanRural         899164 non-null  int64  
 17  RevLineCr          894636 non-null  object 
 18  LowDoc             896582 non-null  object 
 19  ChgOffDate         162699 non-null  object 
 20  DisbursementDate   896796 non-null  object 
 21  DisbursementGross  899164 non-null  object 
 22  BalanceGross       899164 non-null  object 
 23  MIS_Status         897167 non-null  object 
 24  ChgOffPrinGr       899164 non-null  object 
 25  GrAppv             899164 non-null  object 
 26  SBA_Appv           899164 non-null  object 
```

```
dtypes: float64(1), int64(9), object(17)
memory usage: 185.2+ MB
```

In [3]:
```
df.head()
```

Out[3]:

	LoanNr_ChkDgt	Name	City	State	Zip	Bank	BankState	NAICS	App
0	1000014003	ABC HOBBYCRAFT	EVANSVILLE	IN	47711	FIFTH THIRD BANK	OH	451120	
1	1000024006	LANDMARK BAR & GRILLE (THE)	NEW PARIS	IN	46526	1ST SOURCE BANK	IN	722410	
2	1000034009	WHITLOCK DDS, TODD M.	BLOOMINGTON	IN	47401	GRANT COUNTY STATE BANK	IN	621210	
3	1000044001	BIG BUCKS PAWN & JEWELRY, LLC	BROKEN ARROW	OK	74012	1ST NATL BK & TR CO OF BROKEN	OK	0	
4	1000054004	ANASTASIA CONFECTIONS, INC.	ORLANDO	FL	32801	FLORIDA BUS. DEVEL CORP	FL	0	

5 rows × 27 columns

In [4]:
```
df.shape
```

Out[4]:
```
(899164, 27)
```

There are 899164 observations and 27 variables

In [5]:
```
df.columns
```

Out[5]:
```
Index(['LoanNr_ChkDgt', 'Name', 'City', 'State', 'Zip', 'Bank', 'BankState',
       'NAICS', 'ApprovalDate', 'ApprovalFY', 'Term', 'NoEmp', 'NewExist',
       'CreateJob', 'RetainedJob', 'FranchiseCode', 'UrbanRural', 'RevLineCr',
       'LowDoc', 'ChgOffDate', 'DisbursementDate', 'DisbursementGross',
       'BalanceGross', 'MIS_Status', 'ChgOffPrinGr', 'GrAppv', 'SBA_Appv'],
      dtype='object')
```

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

Out[6]:
```
LoanNr_ChkDgt         0
Name                 14
City                 30
State                14
Zip                   0
Bank               1559
BankState          1566
NAICS                 0
ApprovalDate          0
```

```
ApprovalFY               0
Term                     0
NoEmp                    0
NewExist               136
CreateJob                0
RetainedJob              0
FranchiseCode            0
UrbanRural               0
RevLineCr             4528
LowDoc                2582
ChgOffDate          736465
DisbursementDate      2368
DisbursementGross        0
BalanceGross             0
MIS_Status            1997
ChgOffPrinGr             0
GrAppv                   0
SBA_Appv                 0
dtype: int64
```

In [7]:

```
df_new = df.dropna(subset = ["MIS_Status"])
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 897167 entries, 0 to 899163
Data columns (total 27 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   LoanNr_ChkDgt      897167 non-null  int64  
 1   Name               897153 non-null  object 
 2   City               897137 non-null  object 
 3   State              897154 non-null  object 
 4   Zip                897167 non-null  int64  
 5   Bank               895661 non-null  object 
 6   BankState          895654 non-null  object 
 7   NAICS              897167 non-null  int64  
 8   ApprovalDate       897167 non-null  object 
 9   ApprovalFY         897167 non-null  object 
 10  Term               897167 non-null  int64  
 11  NoEmp              897167 non-null  int64  
 12  NewExist           897033 non-null  float64
 13  CreateJob          897167 non-null  int64  
 14  RetainedJob        897167 non-null  int64  
 15  FranchiseCode      897167 non-null  int64  
 16  UrbanRural         897167 non-null  int64  
 17  RevLineCr          892647 non-null  object 
 18  LowDoc             894589 non-null  object 
 19  ChgOffDate         162438 non-null  object 
 20  DisbursementDate   894992 non-null  object 
 21  DisbursementGross  897167 non-null  object 
 22  BalanceGross       897167 non-null  object 
 23  MIS_Status         897167 non-null  object 
 24  ChgOffPrinGr       897167 non-null  object 
 25  GrAppv             897167 non-null  object 
 26  SBA_Appv           897167 non-null  object 
dtypes: float64(1), int64(9), object(17)
memory usage: 191.7+ MB
```

In [8]:

```
pd.isnull(df_new["MIS_Status"]).sum()
```

Out[8]:

```
0
```

In [9]:
```
df_new.loc[:,"SBA_Appv"] = df_new["SBA_Appv"].str.replace(r'^\$|,', '', regex=Tr
df_new["NewExist"] = np.where(df_new["NewExist"] == 0, np.nan, df_new['NewExist'
```

```
/home/nahidanwar/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1
676: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
<ipython-input-9-ebd7bdd10bb1>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
le/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new["NewExist"] = np.where(df_new["NewExist"] == 0, np.nan, df_new['NewExis
t'])
```

In [10]:
```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 897167 entries, 0 to 899163
Data columns (total 27 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   LoanNr_ChkDgt      897167 non-null  int64
 1   Name               897153 non-null  object
 2   City               897137 non-null  object
 3   State              897154 non-null  object
 4   Zip                897167 non-null  int64
 5   Bank               895661 non-null  object
 6   BankState          895654 non-null  object
 7   NAICS              897167 non-null  int64
 8   ApprovalDate       897167 non-null  object
 9   ApprovalFY         897167 non-null  object
 10  Term               897167 non-null  int64
 11  NoEmp              897167 non-null  int64
 12  NewExist           896005 non-null  float64
 13  CreateJob          897167 non-null  int64
 14  RetainedJob        897167 non-null  int64
 15  FranchiseCode      897167 non-null  int64
 16  UrbanRural         897167 non-null  int64
 17  RevLineCr          892647 non-null  object
 18  LowDoc             894589 non-null  object
 19  ChgOffDate         162438 non-null  object
 20  DisbursementDate   894992 non-null  object
 21  DisbursementGross  897167 non-null  object
 22  BalanceGross       897167 non-null  object
 23  MIS_Status         897167 non-null  object
 24  ChgOffPrinGr       897167 non-null  object
 25  GrAppv             897167 non-null  object
 26  SBA_Appv           897167 non-null  float64
dtypes: float64(2), int64(9), object(16)
memory usage: 191.7+ MB
```

In [11]:
```
df_new.shape
```

Out[11]:
```
(897167, 27)
```

Selecting a 25% sample of the data for use in testing

In [12]:
```
rng = np.random.RandomState(20211107)
```

In [13]:
```
test = df_new.sample(frac = 0.25, random_state = rng)
```

In [14]:
```
train_mask = pd.Series(True, index = df_new.index)
train_mask[test.index] = False
train = df_new[train_mask].copy()
train.head()
```

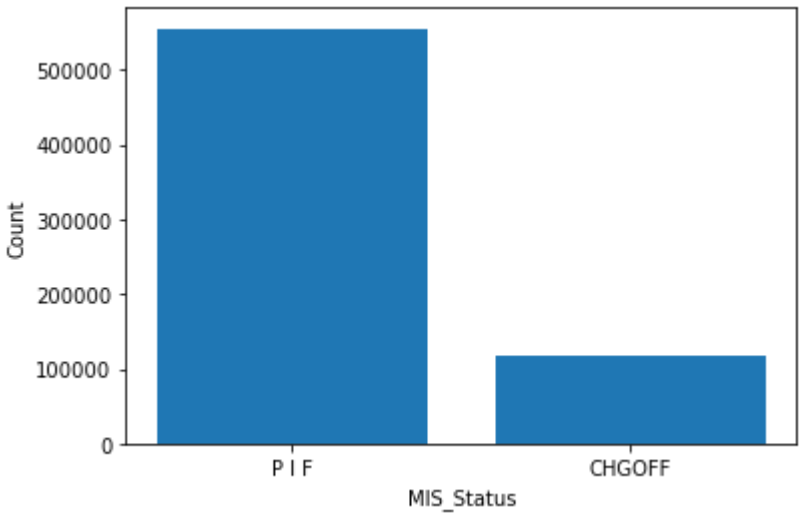
Out[14]:

	LoanNr_ChkDgt	Name	City	State	Zip	Bank	BankState	NAICS	App
0	1000014003	ABC HOBBYCRAFT	EVANSVILLE	IN	47711	FIFTH THIRD BANK	OH	451120	
1	1000024006	LANDMARK BAR & GRILLE (THE)	NEW PARIS	IN	46526	1ST SOURCE BANK	IN	722410	
2	1000034009	WHITLOCK DDS, TODD M.	BLOOMINGTON	IN	47401	GRANT COUNTY STATE BANK	IN	621210	
3	1000044001	BIG BUCKS PAWN & JEWELRY, LLC	BROKEN ARROW	OK	74012	1ST NATL BK & TR CO OF BROKEN	OK	0	
4	1000054004	ANASTASIA CONFECTIONS, INC.	ORLANDO	FL	32801	FLORIDA BUS. DEVEL CORP	FL	0	

5 rows × 27 columns

Describing the distribution of the outcome variable

In [15]:
```
plt.bar(train["MIS_Status"].value_counts().index,train["MIS_Status"].value_count
plt.xlabel("MIS_Status")
plt.ylabel("Count")
plt.show()
```



Here we see that, PIF(Loan is paid) is the majority class. And our goal in this assignment is to predict whether or not the business will pay off their loan.

The accuracy, precision, and recall of the majority-class classifier on the test data:

In [16]:
```
y_estimate = np.ones(test.shape[0])
```

In [17]:
```
y = (test["MIS_Status"] == "P I F").astype(int)
```

In [18]:
```
accuracy_score(y, y_estimate)
```

Out[18]:
```
0.8253169974854208
```

In [19]:
```
precision_score(y, y_estimate)
```

Out[19]:
```
0.8253169974854208
```

our model has a precision of 0.82531 —in other words, when it predicts it is correct 82% of the time

In [20]:
```
recall_score(y, y_estimate)
```

Out[20]:
```
1.0
```

This is a model that produces no false negatives that has a recall of 1.0

Based on our understanding and reading, selecting explanatory variables which are likely to be useful for predicting the outcome.

"SBA_Appv" that is SBA's guaranteed amount of approved loan, is one of the important feature because SBA loans only guarantee a fraction of the total loan sum, if a small business fails on its SBA-guaranteed loan, banks will suffer some losses. As a result, banks confront a difficult decision about whether or not to offer such a loan due to the significant chance of default.

"CreateJob" Number of jobs created,is also a important featurebecause Small businesses have historically been a major source of job creation in the United States; thus, encouraging small business formation and growth offers social advantages by providing jobs and lowering unemployment. The Small Business Administration helps small firms by offering a loan guarantee program, which is designed to encourage banks to lend to small businesses.

"LowDoc" that is LowDoc Loan Program which is categorize as Y = Yes, N = No . Here “LowDoc Loan” program was implemented where loans under $150,000 can be processed using a one-page application. “Yes” indicates loans with a one-page application, and “No” indicates loans with more information attached to the application.

NewExist (1 = Existing Business, 2 = New Business): This represents whether the business is an existing business (in existence for more than 2 years) or a new business (in existence for less than or equal to 2 years). It is surely a important one because a business is new or established (represented as “NewExist” in the dataset) is another potential risk indicator that students identify

Initial Logistic Model

In [21]:
```
x_train = train[["Term", "NoEmp", "SBA_Appv", "CreateJob", "RetainedJob", "LowDo
x_test = test[["Term", "NoEmp", "SBA_Appv", "CreateJob", "RetainedJob", "LowDoc'
y_train = (train["MIS_Status"] == "P I F").astype(int)
y_test = (test["MIS_Status"] == "P I F").astype(int)
```

In [22]:
```
x_train.isnull().sum()
x_test.isnull().sum()
```

Out[22]:
```
Term            0
NoEmp           0
SBA_Appv        0
CreateJob       0
RetainedJob     0
LowDoc        611
NewExist      278
dtype: int64
```

In [23]:
```
x_test.isnull().sum()
```

Out[23]:
```
Term            0
NoEmp           0
SBA_Appv        0
CreateJob       0
RetainedJob     0
LowDoc        611
NewExist      278
dtype: int64
```

In [24]:
```
numeric_variables = ["Term", "NoEmp", "SBA_Appv", "CreateJob", "RetainedJob"]
categorical_variables = ["LowDoc", "NewExist"]
```

In [25]:
```
numeric_processing = Pipeline([("Normalization", StandardScaler())])
```

```
categorical_processing = Pipeline([
    ("Imputer", SimpleImputer(strategy = "most_frequent")),
    ("Onehot", OneHotEncoder(drop = "first", sparse = False, handle_unknown = "e

engineering_pipeline = ColumnTransformer(
transformers = [
    ("Quantitative", numeric_processing, numeric_variables),
    ("Qualitative", categorical_processing, categorical_variables)],remainder= '
```

In [26]:
```
logistic_pipeline = Pipeline([
    ("Feature Engineering", engineering_pipeline),
    ("Classifier", LogisticRegression(penalty = "none", max_iter= 500))])
```

In [27]:
```
logistic_pipeline.fit(x_train, y_train)
```

Out[27]:
```
Pipeline(steps=[('Feature Engineering',
                 ColumnTransformer(transformers=[('Quantitative',
                                                  Pipeline(steps=[('Normalizatio
n',
                                                                   StandardScale
r())]),
                                                  ['Term', 'NoEmp', 'SBA_Appv',
                                                   'CreateJob',
                                                   'RetainedJob']),
                                                 ('Qualitative',
                                                  Pipeline(steps=[('Imputer',
                                                                   SimpleImputer
(strategy='most_frequent')),
                                                                  ('Onehot',
                                                                   OneHotEncoder
(drop='first',

sparse=False))]),
                                                  ['LowDoc', 'NewExist'])])),
                ('Classifier',
                 LogisticRegression(max_iter=500, penalty='none'))])
```

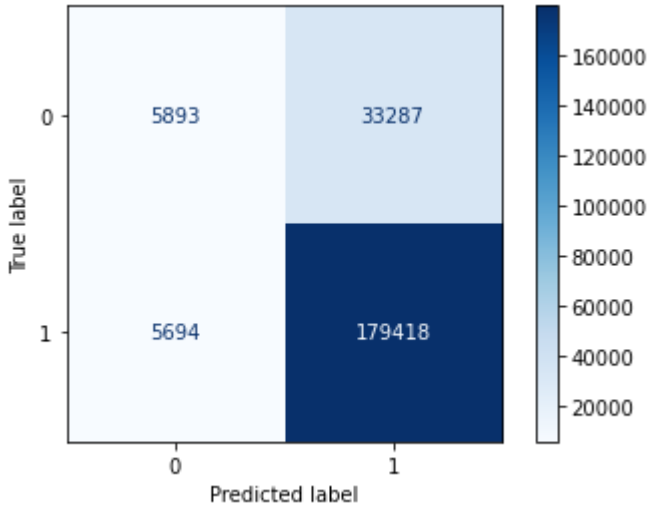
In [28]:
```
pred_test_logistic = logistic_pipeline.predict(x_test)
```

In [29]:
```
accuracy_score(y_test,pred_test_logistic)
```

Out[29]:
```
0.8262042337666969
```

In [30]:
```
plot_confusion_matrix(logistic_pipeline, x_test, y_test, cmap = plt.cm.Blues)
```

Out[30]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f14e471d0d0
>
```



Lasso Regression

In [31]:
```
LASSO_pipeline = Pipeline([
    ("Feature Engineering", engineering_pipeline),
    ("LASSO Classifier", LogisticRegression(penalty = "l1" ,random_state = 0, sc
```

In [32]:
```
LASSO_pipeline.fit(x_train, y_train)
```

Out[32]:
```
Pipeline(steps=[('Feature Engineering',
                 ColumnTransformer(transformers=[('Quantitative',
                                                  Pipeline(steps=[('Normalizatio
n',
                                                                   StandardScale
r())]),
                                                  ['Term', 'NoEmp', 'SBA_Appv',
                                                   'CreateJob',
                                                   'RetainedJob']),
                                                 ('Qualitative',
                                                  Pipeline(steps=[('Imputer',
                                                                   SimpleImputer
(strategy='most_frequent')),
                                                                  ('Onehot',
                                                                   OneHotEncoder
(drop='first',

sparse=False))]),
                                                  ['LowDoc', 'NewExist'])])),
                ('LASSO Classifier',
                 LogisticRegression(penalty='l1', random_state=0,
                                    solver='liblinear'))])
```

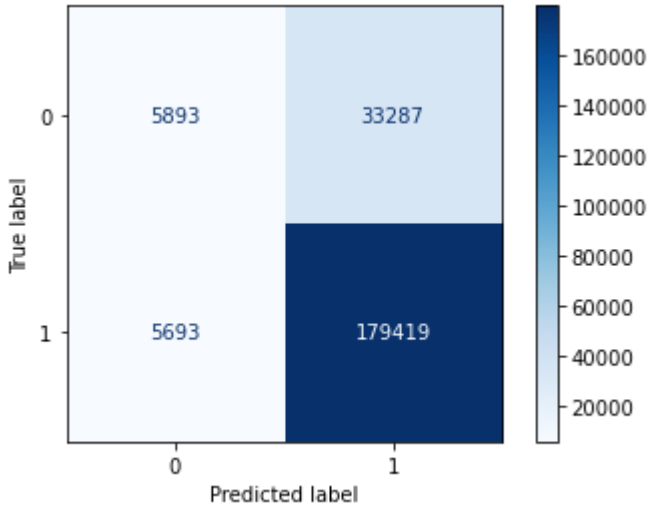
In [33]:
```
pred_test_LASSO = LASSO_pipeline.predict(x_test)
accuracy_score(y_test, pred_test_LASSO)
```

Out[33]:
```
0.8262086922404722
```

In [34]:
```
plot_confusion_matrix(LASSO_pipeline, x_test, y_test, cmap = plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f14b7f2b4f0
```

Out[34]: >



Selecting a set of values for finding the regularization strength for LASSO Regression

In [35]:
```
LASSO_pipeline_cv = Pipeline([
    ("Feature Engineering", engineering_pipeline),
    ("LASSO Classifier", LogisticRegressionCV(Cs = [1, 39, 87], cv = 2, random_s
                                              penalty ='l1', solver = 'liblinear
```

In [36]:
```
LASSO_pipeline_cv.fit(x_train, y_train)
```

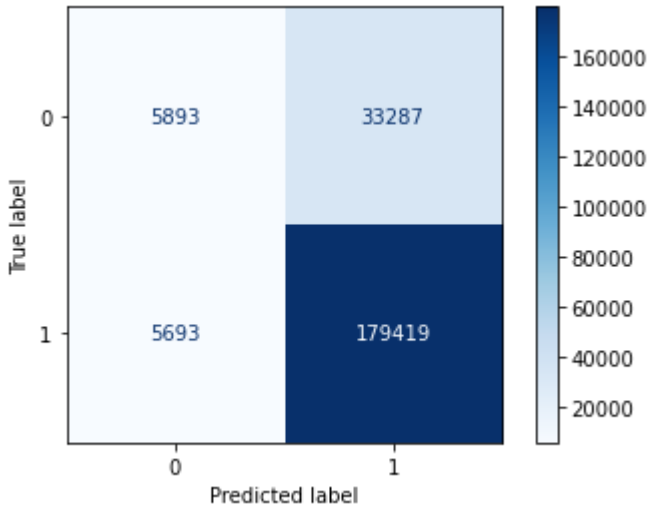
Out[36]:
```
Pipeline(steps=[('Feature Engineering',
                 ColumnTransformer(transformers=[('Quantitative',
                                                  Pipeline(steps=[('Normalizatio
n',
                                                                   StandardScale
r())]),
                                                  ['Term', 'NoEmp', 'SBA_Appv',
                                                   'CreateJob',
                                                   'RetainedJob']),
                                                 ('Qualitative',
                                                  Pipeline(steps=[('Imputer',
                                                                   SimpleImputer
(strategy='most_frequent')),
                                                                  ('Onehot',
                                                                   OneHotEncoder
(drop='first',

sparse=False))]),
                                                  ['LowDoc', 'NewExist'])])),
                ('LASSO Classifier',
                 LogisticRegressionCV(Cs=[1, 39, 87], cv=2, penalty='l1',
                                      random_state=0, solver='liblinear'))])
```

In [37]:
```
pred_test_LASSO_cv = LASSO_pipeline_cv.predict(x_test)
accuracy_score(y_test, pred_test_LASSO_cv)
```

Out[37]:
```
0.8262086922404722
```

In [38]:
```
plot_confusion_matrix(LASSO_pipeline_cv, x_test, y_test, cmap = plt.cm.Blues)
```

Out[38]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f14b7f61a00
>
```



ElasticNet

In [39]:
```
ELASTICNET_pipeline = Pipeline([
    ("Feature Engineering", engineering_pipeline),
    ("ELASTICNET Classifier", LogisticRegression(penalty = "elasticnet" ,random_
                                                 solver = "saga", l1_ratio = 0.5
```

In [40]:
```
ELASTICNET_pipeline.fit(x_train, y_train)
```

```
/home/nahidanwar/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sa
g.py:328: ConvergenceWarning: The max_iter was reached which means the coef_ did
not converge
  warnings.warn("The max_iter was reached which means "
```

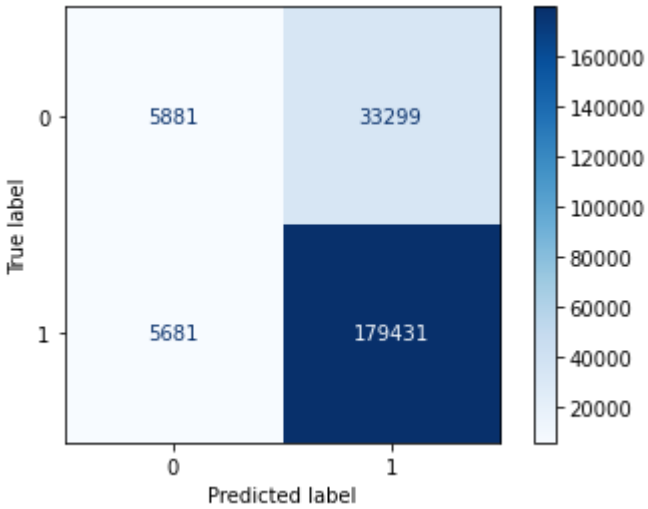
Out[40]:
```
Pipeline(steps=[('Feature Engineering',
                 ColumnTransformer(transformers=[('Quantitative',
                                                  Pipeline(steps=[('Normalizatio
n',
                                                                   StandardScale
r())]),
                                                  ['Term', 'NoEmp', 'SBA_Appv',
                                                   'CreateJob',
                                                   'RetainedJob']),
                                                 ('Qualitative',
                                                  Pipeline(steps=[('Imputer',
                                                                   SimpleImputer
(strategy='most_frequent')),
                                                                  ('Onehot',
                                                                   OneHotEncoder
(drop='first',

sparse=False))]),
                                                  ['LowDoc', 'NewExist'])])),
                ('ELASTICNET Classifier',
                 LogisticRegression(l1_ratio=0.5, penalty='elasticnet',
                                    random_state=0, solver='saga'))])
```

In [41]:
```
pred_test_ELASTICNET = ELASTICNET_pipeline.predict(x_test)
accuracy_score(y_test, pred_test_ELASTICNET)
```

Out[41]:
```
0.8262086922404722
```

In [42]:
```
plot_confusion_matrix(ELASTICNET_pipeline, x_test, y_test, cmap = plt.cm.Blues)
```

Out[42]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f149e9a5220
>
```



Random Forest

In [43]:
```
RF_pipeline = Pipeline([
    ("Feature Engineering", engineering_pipeline),
    ("Random Forrest", RandomForestClassifier(max_depth = 10, random_state = 0,
```

In [44]:
```
RF_pipeline.fit(x_train, y_train)
```

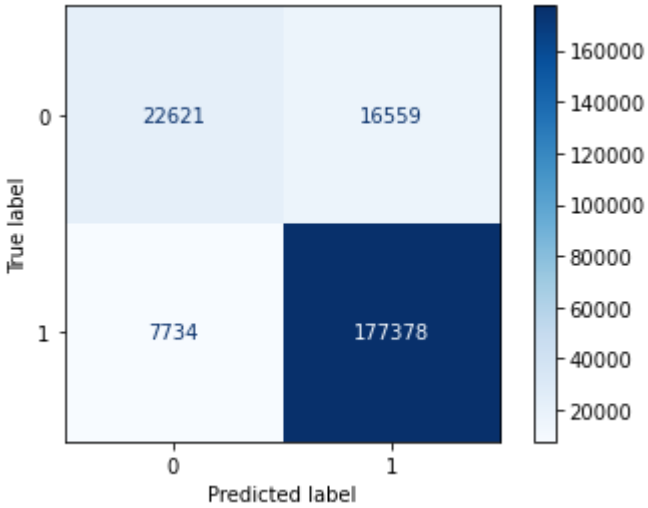
Out[44]:
```
Pipeline(steps=[('Feature Engineering',
                 ColumnTransformer(transformers=[('Quantitative',
                                                  Pipeline(steps=[('Normalizatio
n',
                                                                   StandardScale
r())]),
                                                  ['Term', 'NoEmp', 'SBA_Appv',
                                                   'CreateJob',
                                                   'RetainedJob']),
                                                 ('Qualitative',
                                                  Pipeline(steps=[('Imputer',
                                                                   SimpleImputer
(strategy='most_frequent')),
                                                                  ('Onehot',
                                                                   OneHotEncoder
(drop='first',

sparse=False))]),
                                                  ['LowDoc', 'NewExist'])])),
                ('Random Forrest',
                 RandomForestClassifier(max_depth=10, random_state=0))])
```

In [45]:
```
pred_test_RF = RF_pipeline.predict(x_test)
accuracy_score(y_test, pred_test_RF)
```

Out[45]: 0.8916902965776755

In [46]:
```
plot_confusion_matrix(RF_pipeline, x_test, y_test, cmap = plt.cm.Blues)
```

Out[46]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f14b6457c40>



Final summary and Reflection

In this assignment, we get to learn how data science play a vital role in Economical problems also. It is a rich dataset for educators. In this current world vast amount of data is available, which is so valuable for measuring performance and building up a solid evidence base before making major decisions that influence the direction of a business,economics especially in loan issuing decision.

Our final recommendation is that loan(s) could be approved. We really get to know problems of economics can be treated in a more scientific way .This assignment is a great source to learn a great range of statistical concepts and highlight how data can be used to inform real business decisions.