Loan Approval for Small Businesses

CONTENT

PROJECT OVERVIEW

DATASET

Small Business Loan Status.

Source: U.S. Small Business Administration (SBA).
Format: **.csv**

https://www.kaggle.com/datasets/mirbektoktogaraev/should-this-loan-be-approved-or-denied

PROBLEM STATEMENT

Should the loan be approved or denied?

Predict a possible future **status of a loan** (Paid in full OR Charged off).
If the predicted status of a loan is “Charged off” - granting it may be risky.

DATASET STRUCTURE

27 columns x 899,164 rows of data.

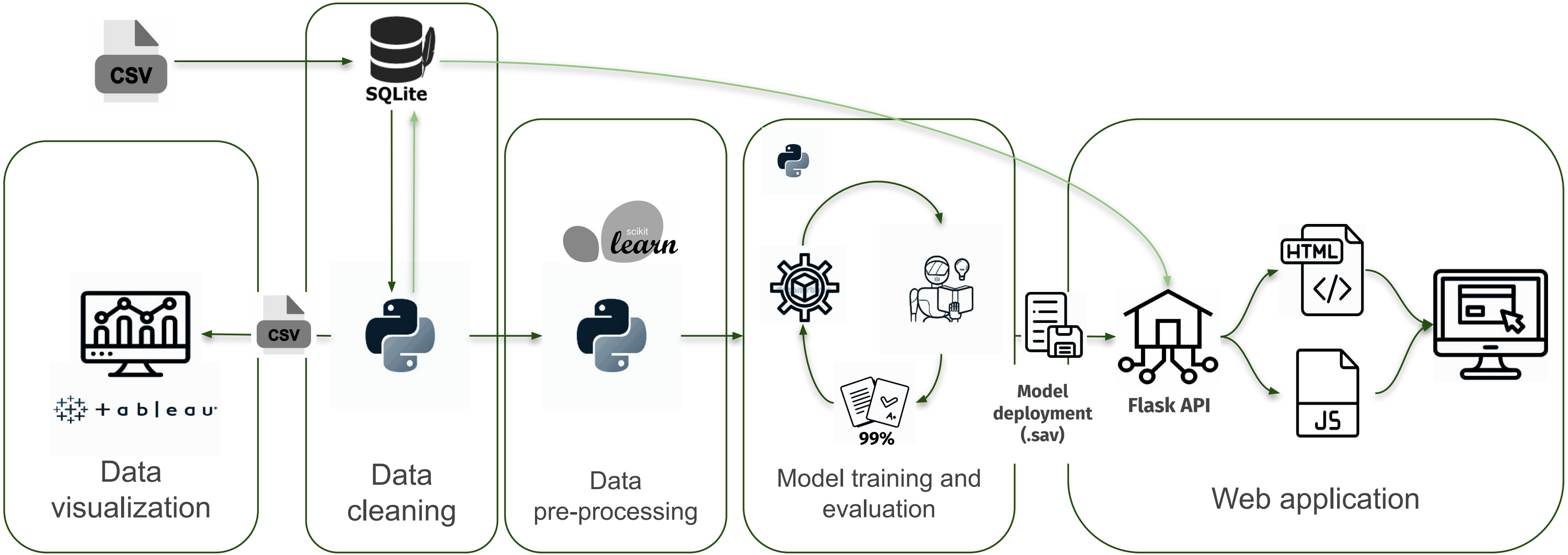
Data types: categorical and continuous data (str, int/float, date).

Steps for data cleaning: replacing categorical missing values with 0, dropping boolean missing values.

MOTIVATION

Model should help **banks** predict whether an applicant company will be able to pay off the loan, based on the provided information about company’s size, location, term of loan etc.

SOLUTION ARCHITECTURE



DATA CLEANING AND PREPROCESSING

a. The CSV file was loaded into a relational database (SQLITE) and then loaded into a jupyter notebook to clean the data.

b. The columns were transformed to their respective type (string, integer, float) and unwanted columns were dropped.

c. The goal was to create two separate dataframes :- i) one for Tableau visualization ii) the other for machine learning , where only the required columns (independent variables) helped in predicting the dependent variable

d. PANDAS was used to clean data and Tableau tools were used to filter and visualize the data.

DATA CLEANING AND PREPROCESSING

e. . For Machine Learning to transform the data we used Column Transformer, Standard Scaler and train_test_split to validate the data for feeding it into the different models created and into the Neural Networks.

f. We then used the pickle function to save the train model and load it in FLASK to predict outcome from user inputted values.

g. The Flask file also filtered data to help JS file read data from and plot maps accordingly.

MACHINE LEARNING

MACHINE LEARNING - Leon/Makram

a. Once the dataset was cleaned and new dataframe was created with only the necessary columns that were relevant to the Machine Building Model we implement the dataframe into different ML models.

The dataframe was trained was trained, tested and validated on different models.

We then created another testing data by taking randomly 25% of the dataset to see if the Recall and Accuracy holds good.

The following models were used:

1. Random Forest Classifier
2. Voting Classifier (consisting of Logistic Regression, Decision Tree classifier and Random Forest Classifier
3. Bagging Classifier with base estimator as Decision Tree Classifier.

MACHINE LEARNING

MACHINE LEARNING

All 3 Models showed accuracy well above 80% . We however decided to go with Bagging Classifier as the model to be deployed as the model gives out low bias and low variance

Variance, in the context of Machine Learning, is a type of error that occurs due to a model's sensitivity to small fluctuations in the training set/new data being introduced, in bagging this new data is then partly fitted into new each model hence resulting in low variance.

```python
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=500, random_state=78)

# Fitting the model
rf_model = rf_model.fit(X_train_scaled, y_train)
predictions_rf = rf_model.predict(X_test_scaled)
print(classification_report(y_test, predictions_rf))
```

✓ 4m 51.2s

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     21431
           1       0.88      0.83      0.85      7312

    accuracy                           0.93     28743
   macro avg       0.91      0.89      0.90     28743
weighted avg       0.93      0.93      0.93     28743
```

```python
predictions_rf_validate = rf_model.predict(X_val_scaled)
print(classification_report(y_val, predictions_rf_validate))
```

✓ 5.3s

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     23812
           1       0.88      0.83      0.85      8125

    accuracy                           0.93     31937
   macro avg       0.91      0.90      0.90     31937
weighted avg       0.93      0.93      0.93     31937
```

VOTING CLASSIFIER

```
#Voting Classifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import balanced_accuracy_score
rfc = RandomForestClassifier(random_state=42)
dtc = DecisionTreeClassifier(random_state=42)
lr = LogisticRegression()
```

✓ 0.0s

```
pipe = VotingClassifier([('dtc', dtc),('rfc', rfc),('lr', lr)], weights = [4,5,1])
pipe.fit(X_train_scaled, y_train)

y_predict = pipe.predict(X_25_scaled)
print(balanced_accuracy_score(y_25, y_predict))

print(classification_report(y_25, y_predict))
```

✓ 1m 4.5s

```
0.9759983800672256
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     59698
           1       0.98      0.96      0.97     20143

    accuracy                           0.98     79841
   macro avg       0.98      0.98      0.98     79841
weighted avg       0.98      0.98      0.98     79841
```

Bagging Classifier

```python
#Using Bagging
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn .svm import SVC
bag = BaggingClassifier(
    base_estimator = DecisionTreeClassifier(),
    n_estimators = 500,
    max_samples = 0.60, #no of samples from Xtrain
    bootstrap = True,
    random_state =  42
)
```

✓ 0.0s

```python
bag.fit(X_train_scaled, y_train)

y_pred = bag.predict(X_test_scaled)
y_predict_25 = bag.predict(X_25_scaled)
print("Classification report for X_Test")
print(classification_report(y_test, y_pred))

print("Classification report for X_25")
print(classification_report(y_25, y_predict_25))
```

✓ 14m 19.7s

```
c:\Users\ljm47\anaconda3\envs\dev\lib\site-packages\sklearn\ensemble\_base.p
  warnings.warn(
Classification report for X_Test
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     21431
           1       0.88      0.88      0.88      7312

    accuracy                           0.94     28743
   macro avg       0.92      0.92      0.92     28743
weighted avg       0.94      0.94      0.94     28743
```

```python
bag_N.fit(X_train_scaled_N, y_train_N)

y_pred = bag_N.predict(X_test_scaled_N)
y_predict_val = bag_N.predict(X_val_scaled_N)
print("Classification report for X_Test_N")
print(classification_report(y_test_N, y_pred))

print(classification_report(y_val_N,y_predict_val))
```

✓ 4m 44.7s

```
c:\Users\ljm47\anaconda3\envs\dev\lib\site-packages\sklearn\ensembl
  warnings.warn(
Classification report for X_Test_N
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     38100
           1       0.87      0.88      0.88     12999

    accuracy                           0.94     51099
   macro avg       0.92      0.92      0.92     51099
weighted avg       0.94      0.94      0.94     51099

              precision    recall  f1-score   support

           0       0.96      0.95      0.96     47624
           1       0.87      0.88      0.87     16249

    accuracy                           0.94     63873
   macro avg       0.91      0.92      0.92     63873
weighted avg       0.94      0.94      0.94     63873
```

NEURAL NETWORKS WITH KERAS TUNER USED ON 75% OF THE DATASET

```
#Using the best hyperparameters and building a model and fitting the Xtrain and Ytrain on the model

tuner_75.get_best_hyperparameters()[0].values
model = tuner_75.get_best_models(num_models=1)[0]
model.fit(X_train_scaled, y_train, epochs= 50)
```

✓ 15m 3.7s

```
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
model.evaluate(X_test_scaled, y_test, verbose = 2)
```

✓ 1.6s

```
899/899 - 1s - loss: 0.2687 - accuracy: 0.8836 - 1s/epoch - 2ms/step

[0.2687359154224396, 0.8836238384246826]
```

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_flatten))
```

✓ 0.1s

	precision	recall	f1-score	support
0	0.92	0.93	0.92	21431
1	0.78	0.75	0.77	7312
accuracy			0.88	28743
macro avg	0.85	0.84	0.84	28743
weighted avg	0.88	0.88	0.88	28743

TESTING THE REMAINING 25% OF DATASET ON THE “model” object

```
from sklearn.metrics import classification_report

print(classification_report(y_25, y_flatten25))
```

✓ 0.2s

	precision	recall	f1-score	support
0	0.92	0.93	0.93	59698
1	0.79	0.76	0.78	20143
accuracy			0.89	79841
macro avg	0.85	0.85	0.85	79841
weighted avg	0.89	0.89	0.89	79841

We will now input data in training model to show how the model predicts Data.

127.0.0.1:5000/predict

ApprovalDate	Date/Time	Date SBA commitment issued
ApprovalFY	Text	Fiscal year of commitment
Term	Number	Loan term in months
NoEmp	Number	Number of business employees
NewExist	Text	1 = Existing business, 2 = New business
CreateJob	Number	Number of jobs created
RetainedJob	Number	Number of jobs retained
FranchiseCode	Text	Franchise code, (00000 or 00001) = No franchise
UrbanRural	Text	1 = Urban, 2 = rural, 0 = undefined
RevLineCr	Text	Revolving line of credit: Y = Yes, N = No
LowDoc	Text	LowDoc Loan Program: Y = Yes, N = No
ChgOffDate	Date/Time	The date when a loan is declared to be in default
DisbursementDate	Date/Time	Disbursement date
DisbursementGross	Currency	Amount disbursed
BalanceGross	Currency	Gross amount outstanding
MIS_Status	Text	Loan status charged off = CHGOFF, Paid in full = PIF
ChgOffPrinGr	Currency	Charged-off amount
GrAppv	Currency	Gross amount of loan approved by bank
SBA_Appv	Currency	SBA's guaranteed amount of approved loan

CA

Term

NoEmp

Existing Business

CreateJob

RetainedJob

Is it a Franchise: NO

Is it a Urban or Rural : Urban

Is it a Revolvinsg Credit : NO

Is it having Low Documentation : NO

GrAppv

Retail_trade

Predict

the Loan Amount code [0]. If Code is 0 , Client likely to NOT DEFAULT If Code = 1, Client is like to default, Proceed with caution

Go to Dashboard Go to Tableau Dashboard for more details

DATA VISUALIZATION - Tableau - interactive

CONCLUSION

SUMMARY

- Neural network didn’t show high accuracy in classification problem such as loan status prediction.
- Combination of multiple weaker classifiers showing better accuracy results.
- Having a validation dataset is important for evaluating the model performance.

LIMITATIONS

- Records without information in target variable column were removed.
- Records in critical columns with categorical values of type boolean were removed where values were missing.
- Visualization is connected to a static data source.

NEXT STEPS

- Explore options to move data storage solution to Cloud.

- The project can be further expanded to build a model for the SBA use, to identify the insurance amount, that should be granted, to allow to mitigate risks for the organization.