Predicting Loan Applicant Risk Profile Using Machine Learning

Commerce 3FN3 Big Data in Finance

G-15

Problem Background

Every year banks receive thousands of loan applications each with unique circumstances that must be evaluated.

This presents a variety of issues for banks to solve such as time consumption, inconsistencies, limited scalability, underutilization of data, and compliance risks.

Without the support of software and machine learning, interacting with all the information about each loan applicant is impossible for even a large team of employees.

Banks such as Deutsche Bank, Bank of America, and JP Morgan Chase are adopting machine learning and robotic process automation to improve their loan processing systems. They are using software platforms that can streamline the loan processing workflow such as nCino, LendingPad, Finastra, and Fiserv

The next step for a definite solution to automating banks loans is combining the features of the existing systems into a streamlined and accurate program. This project will provide an accurate, transparent, and simple system that highlights the risk of each loan applicant to assist institutions of all sizes in the decision-making process.

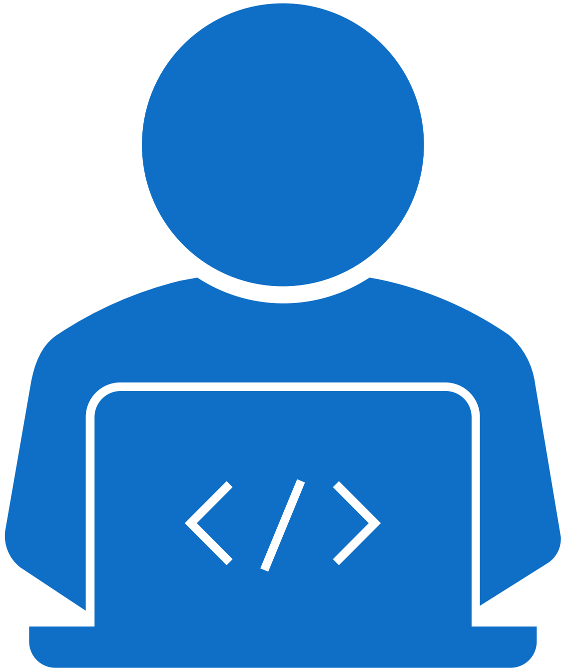
Project Objective

- The algorithm itself, given client metrics, will evaluate the applicant based on previous loan applications from other clients and their outcomes
- This evaluation will determine:

1. if the client is likely or unlikely to successfully pay off the loan
2. if they should be approved or declined for the loan

Proposed Solution

Our solution takes in a large dataset of financial profiles, cleans it, and then uses supervised learning to train an algorithm that can determine the risk of a loan and whether it should be approved. This solution provides a fast and accurate method for banks to evaluate loan applications at a large scale.



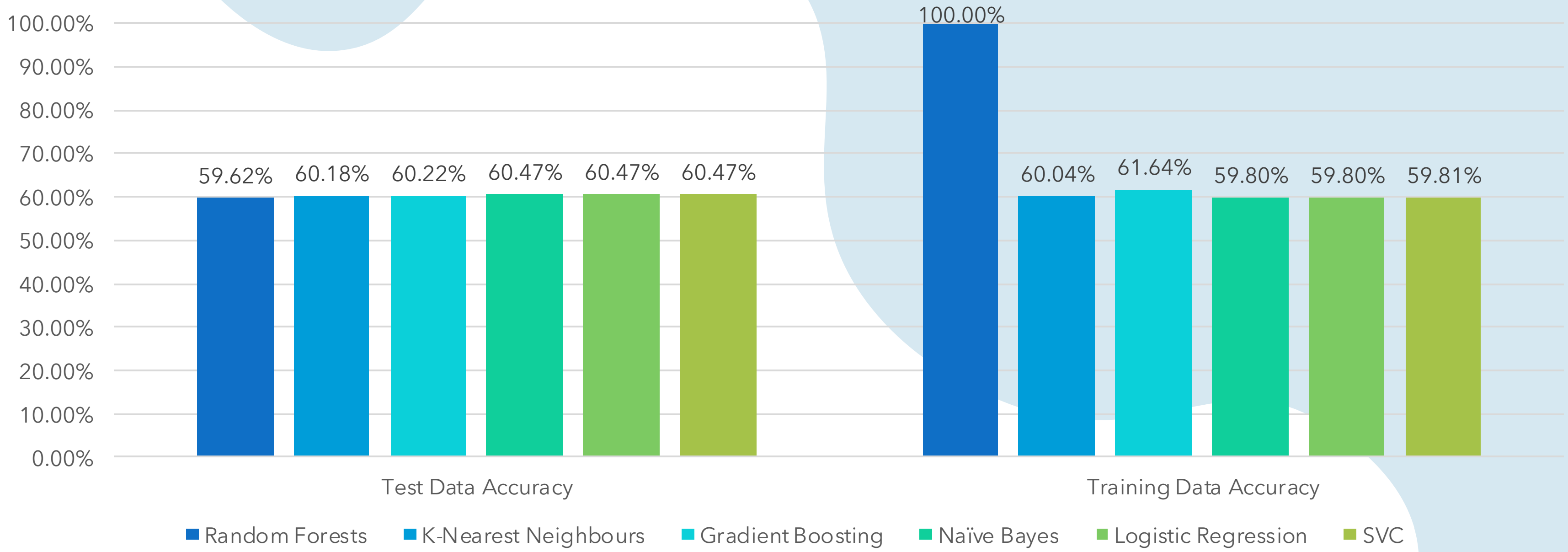
Dataset

- Data is to be gathered from financial institution databases of prior applications, then cleaned and normalized
- Datasets on the side are examples with 15000 rows and 19 columns of client data before and after being processed

	Age	Gender	Education Level	Marital Status	Income	Credit Score	Loan Amount	Loan Purpose	Employment Status	Years at Current Job	Payment History	Debt-to-Income Ratio	Assets Value	Number of Dependents	City	State	Country	Previous Defaults	Marital Status Change	Risk Rating
0	49	Male	PhD	Divorced	72799	688	45713	Business	Unemployed	19	Poor	0.154313	120228	0	Port Elizabeth	AS	Cyprus	2	2	Low
1	57	Female	Bachelor's	Widowed	NaN	690	33835	Auto	Employed	6	Fair	0.14892	55849	0	North Catherine	OH	Turkmenistan	3	2	Medium
2	21	Non-binary	Master's	Single	55687	600	36623	Home	Employed	8	Fair	0.362398	180700	3	South Scott	OK	Luxembourg	3	2	Medium
3	59	Male	Bachelor's	Single	26508	622	26541	Personal	Unemployed	2	Excellent	0.454964	157319	3	Robinhaven	PR	Uganda	4	2	Medium
4	25	Non-binary	Bachelor's	Widowed	49427	766	36528	Personal	Unemployed	10	Fair	0.143242	287140	NaN	New Heather	IL	Namibia	3	1	Low
5	30	Non-binary	PhD	Divorced	NaN	717	15613	Business	Unemployed	5	Fair	0.295984	NaN	4	Brianland	TN	Iceland	3	1	Medium
6	31	Non-binary	Master's	Widowed	45280	672	6553	Personal	Self-employed	1	Good	0.37889	NaN	NaN	West Lindaview	MD	Bouvet Island (Bouvetoya)	0	1	Low
7	18	Male	Bachelor's	Widowed	93678	NaN	NaN	Business	Unemployed	10	Poor	0.396636	246597	1	Melissahaven	MA	Honduras	1	1	Low
8	32	Non-binary	Bachelor's	Widowed	20205	710	NaN	Auto	Unemployed	4	Fair	0.335965	227599	0	North Beverly	DC	Pitcairn Islands	4	2	Low
9	55	Male	Bachelor's	Married	32190	600	29918	Personal	Self-employed	5	Excellent	0.484333	130507	4	Davidstad	VT	Thailand	NaN	2	Low

	Age	Gender	Education Level	Marital Status	Income	Credit Score	Loan Amount	Loan Purpose	Employment Status	Years at Current Job	Payment History	Debt-to-Income Ratio	Assets Value	Number of Dependents	Previous Defaults	Marital Status Change	Risk Rating
0	49	Male	PhD	Divorced	72799	688	45713	Business	Unemployed	19	Poor	0.154313	120228	0	2	2	Low
1	57	Female	Bachelor's	Widowed	69773	690	33835	Auto	Employed	6	Fair	0.14892	55849	0	3	2	Medium
2	21	Non-binary	Master's	Single	55687	600	36623	Home	Employed	8	Fair	0.362398	180700	3	3	2	Medium
3	59	Male	Bachelor's	Single	26508	622	26541	Personal	Unemployed	2	Excellent	0.454964	157319	3	4	2	Medium
4	25	Non-binary	Bachelor's	Widowed	49427	766	36528	Personal	Unemployed	10	Fair	0.143242	287140	2	3	1	Low
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14995	23	Non-binary	Bachelor's	Widowed	48088	609	26187	Home	Self-employed	2	Fair	0.317633	159362	4	2	0	Low
14996	56	Male	PhD	Single	107193	700	35111	Auto	Self-employed	10	Fair	0.155126	79102	2	0	0	Medium
14997	29	Non-binary	PhD	Married	46250	642	44369	Home	Unemployed	19	Excellent	0.593999	196930	4	2	1	High
14998	53	Non-binary	PhD	Divorced	40180	638	32752	Home	Self-employed	12	Excellent	0.478035	276060	2	0	2	High
14999	24	Non-binary	Bachelor's	Widowed	69773	765	27544	Personal	Self-employed	18	Excellent	0.116083	71699	3	3	2	Low

Summary of Results



- The test accuracies of all algorithms is relatively low which is indicative of a dataset issue
- For an actual implementation, good and strongly correlated data or more datapoints would be necessary for higher accuracy

Solution Details

Used Libraries: pandas, sklearn

Random Forests

```
random_forests_pipe = Pipeline(
 [
 ('transform_columns', ColumnTransformation),
 ('pca', PCA(n_components = 32)),
 ('rf', RandomForestClassifier(random_state = 100))
 ]
)

random_forests_pipe.fit(X_train, Y_train)
print("Random Forests Accuracy(test data):", random_forests_pipe.score(X_test, Y_test) * 100, "%")
print("Random Forests Accuracy(training data):", random_forests_pipe.score(X_train, Y_train) * 100, "%")
```

K-Nearest Neighbours

```
KNN_pipe = Pipeline(
 [
 ('transform_columns', ColumnTransformation),
 ('pca', PCA(n_components = 32)),
 ('knn', KNeighborsClassifier(n_neighbors=5))
 ]
)

KNN_pipe.fit(X_train, Y_train)
print("K-Nearest Neighbours Accuracy(test data):", KNN_pipe.score(X_test, Y_test) * 100, "%")
print("K-Nearest Neighbours Accuracy(training data):",KNN_pipe.score(X_train, Y_train) * 100, "%")
```

Gradient Boosting

```
gradient_boosting_pipe = Pipeline(
 [
 ('transform_columns', ColumnTransformation),
 ('pca', PCA(n_components = 32)),
 ('gb', GradientBoostingClassifier())
 ]
)

gradient_boosting_pipe.fit(X_train, Y_train)
print("Gradient Boosting Accuracy(test data):", gradient_boosting_pipe.score(X_test, Y_test) * 100, "%")
print("Gradient Boosting Accuracy(training data):", gradient_boosting_pipe.score(X_train, Y_train) * 100, "%")
```

Naive Bayes

```
naive_bayes_pipe = Pipeline(
 [
 ('transform_columns', ColumnTransformation),
 ('pca', PCA(n_components = 32)),
 ('nb', GaussianNB())
 ]
)

naive_bayes_pipe.fit(X_train, Y_train)
print("Naive Bayes Accuracy(test data):", naive_bayes_pipe.score(X_test, Y_test) * 100, "%")
print("Naive Bayes Accuracy(training data): ", naive_bayes_pipe.score(X_train, Y_train) * 100, "%")
```

Logistic Regression

```
logistic_regression_pipe = Pipeline(
 [
 ('transform_columns', ColumnTransformation),
 ('pca', PCA(n_components = 32)),
 ('logReg', linear_model.LogisticRegression())
 ]
)

logistic_regression_pipe.fit(X_train, Y_train)
print("Logistic Regression Accuracy(test data):", logistic_regression_pipe.score(X_test, Y_test) * 100, "%")
print("Logistic Regression Accuracy(training data):", logistic_regression_pipe.score(X_train, Y_train) * 100, "%")
```

SVC

```
svc_pipe = Pipeline(
 [
 ('transform_columns', ColumnTransformation),
 ('pca', PCA(n_components = 32)),
 ('svc', SVC())
 ]
)

svc_pipe.fit(X_train, Y_train)
print("SVC Accuracy(test data):", svc_pipe.score(X_test, Y_test) * 100, "%")
print("SVC Accuracy(training data):", svc_pipe.score(X_train, Y_train) * 100, "%")
```

Recommendations and Business Impacts

Recommendation

<u>Implement a machine learning algorithm to assist in loan application processing</u>

- Will allow for more flexible reaction time and capacity for the processing of new applications.
- No longer need to rely on employee capacity for tasks which can be automated.

Impacts

<u>Employee Reduction</u>

- With the machine learning algorithm implemented, fewer man hours to process loan applications will be required

<u>Faster loan application turn around times</u>

- Algorithm can operate around the clock or be scaled easily as demand fluctuates, application backlog can be reduced or eliminated completely

Timeline

Months 1-3
Establish the exact type of data set and parameters upon which the algorithm will be trained on and collect all data into raw dataset

Months 4-6
Determine optimal clean up and normalization script for the raw dataset

Months 7-10
Test all algorithms with varying parameters to identify which is most accurate

Months 11-13
Implement policies for future data collection and processing

Months 14-18
Create a front end to the algorithm for the employees who will be utilizing the tool

Months 19-31
Begin solution rollout and continually test and make changes as needed based on feedback

Ongoing
Implement full solution and reduce staff as needed

Conclusion – What we learned

Conclusion

- By implementing a machine learning algorithm, tasks that typically take a significant amount of time can be automated. This allows for operating costs to be reduced while increasing overall flexibility as it is easier to scale compute power than employee count.

What we learned

- Developed a deeper understanding of how to implement machine learning to real life applications
- Learned of the limitations of machine learning and the importance of large datasets to train the algorithms upon
- Became more familiar with each different machine learning model and their parameters, developing an understanding of the importance of fine-tuning algorithms based on their application