DBA3751 Independent Study in Business Analytics

Neural Networks in Quantitative Finance

Final Report

Name	Matriculation Number
Chern Tat Sheng	A0183739N

Date: 21 Apr 2023

1. Introduction

This report aims to provide a comprehensive overview of the research, implementation and evaluation of neural network models applied to several pillars in quantitative finance, mainly in optimising the volatility surface, option pricing and classification for credit lending.

In my analysis, I will be focusing on issues in 3 main pillars. Each pillar will contain sections where I will go into detail for exploratory data analysis, pre-processing and finally testing of each neural network model in different areas of quantitative finance, in predicting for implied volatility, option pricing and credit lending.

2. Research methodology

2.1 Overview

Within each pillar, I will utilize data from traditional derivative markets (VIX and S&P 500 Options) and/or non-traditional markets (Bitcoin Options). For each dataset, I will randomly employ a $P : 1$ split (P: number of parameters to be predicted) into the training and test sets, to improve result reliability (Roshan, 2022). A neural network model for each pillar will be fitted using the applicable training set.

To sieve out the best performing model, I will compare the fitted and tuned neural network models against other conventional machine learning models, using applicable performance measures per research pillar.

Research Pillar	Models for Comparison	Performance Measures	Dataset
Volatility Surface	Neural Networks, Linear Regression	Mean Squared Error (MSE), Gain Ratio on MSE	Options Data, VIX Dataset
Option Pricing	Neural Networks	Mean Absolute Error (MAE)	BTC and ETH Options Dataset
Credit Lending	Neural Networks, Logistic Regression, Random Forest, XGBoost Classifier, Ensemble (LR, RF, XG)	AUC Score, F1 Score and Precision Score	2008 - 2021 Credit Lending Dataset

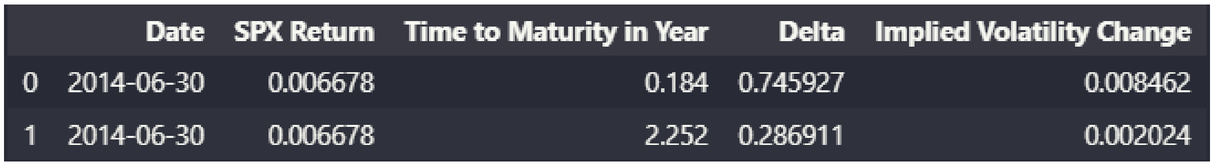
Table 1. Overview of models and data

3. Independent research

3.1 Implied volatility with neural networks

3.1.1 Exploratory data analysis and pre-processing

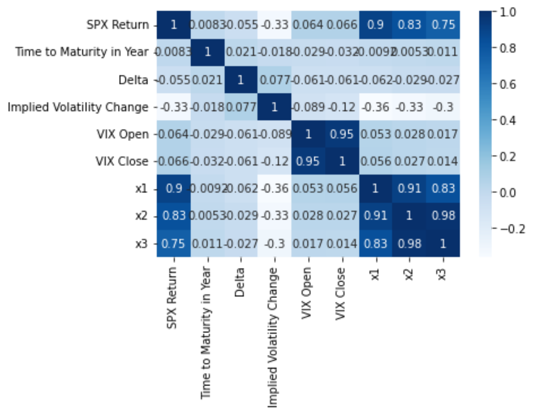
The estimation of implied volatility from options data is a complex and critical task in the field of financial analytics. Neural networks have shown tremendous potential in modeling and predicting complex financial data. In this study, I leverage the power of neural networks to develop a robust model for predicting the implied volatility of the options data below. To do so, I conducted a comparative analysis of an untuned neural network model against a basic linear regression model to compare the performance of each model.



	Date	SPX Return	Time to Maturity in Year	Delta	Implied Volatility Change
0	2014-06-30	0.006678	0.184	0.745927	0.008462
1	2014-06-30	0.006678	2.252	0.286911	0.002024

Figure 1. Options Dataset with headers

In tackling this question, the Hull and White (Kenton, 2022) model below was suggested for its simplicity in Linear Regression models, where R is the return on the asset (Daily), T is the option's time to maturity and the delta measure of moneyness of the option, with a, b and c being constants. Hull estimates a, b and c by regressing implied volatility changes with against $R/\sqrt{T}$, $R\delta/\sqrt{T}$, and $R\delta\ 2/\sqrt{T}$. His simple model involved the creation of 3 features for the linear regression: **x1**: R, **x2**: R / (T*δ) and **x3**: x2 * δ . and I demonstrated the efficacy of incorporating the volatility index VIX as a feature to improve the model's performance.



	SPX Return	Time to Maturity in Year	Delta	Implied Volatility Change	VIX Open	VIX Close	x1	x2	x3
SPX Return	1	0.0083	-0.055	-0.33	0.064	0.066	0.9	0.83	0.75
Time to Maturity in Year	0.0083	1	0.021	-0.018	-0.029	-0.032	-0.0092	0.0053	0.011
Delta	-0.055	0.021	1	0.077	-0.061	-0.061	-0.062	-0.029	-0.027
Implied Volatility Change	-0.33	-0.018	0.077	1	-0.089	-0.12	-0.36	-0.33	-0.3
VIX Open	0.064	-0.029	-0.061	-0.089	1	0.95	0.053	0.028	0.017
VIX Close	0.066	-0.032	-0.061	-0.12	0.95	1	0.056	0.027	0.014
x1	0.9	-0.0092	-0.062	-0.36	0.053	0.056	1	0.91	0.83
x2	0.83	0.0053	-0.029	-0.33	0.028	0.027	0.91	1	0.98
x3	0.75	0.011	-0.027	-0.3	0.017	0.014	0.83	0.98	1

Figure 2. Correlation between features

In my initial exploration, I conducted a correlation analysis, and visualized the data distribution in the plot below. From this plot, I observed that the correlation between x1, x2 and x3 are highly correlated, posing a great risk of multicollinearity. To investigate this finding, I performed the linear regression once with x1, x2 and x3 and compared the results against a linear regression performed with R, T and δ to obtain the scores in Table 2.

Test Loss (MSE) for John Hulls Simple Model	7.632364938220805e-05
Test Loss (MSE) without Model	7.688608879017142e-05

Table 2. Performance scores for Simple Model vs No model

With this simple model, the MSE loss saw an improvement of close to 1%. Contrary to our preconceived notion of multicollinearity. Engineered features are made by combining multiple original attributes in a way that creates a new feature, which captures a specific aspect of the dataset, which may not have been present before. These features are typically created with domain knowledge and intuition, and are designed to provide new and meaningful information to the model. Since engineered features are constructed to capture specific information and are not just simple transformations of existing features, they are less likely to suffer from multicollinearity, which occurs when two or more independent variables are highly correlated with each other. This further reduces the risk of multicollinearity in the model. Thus, for our purposes, I will not be removing them from the dataset.

To prepare the data for analysis, I implemented several preprocessing techniques, including feature scaling and missing data handling. Subsequently, I will split the data into training and testing sets, employing a $P : 1$ split (P: number of parameters to be predicted) into the training and test sets, to improve result reliability (Roshan, 2022).

3.1.2 Performance measures

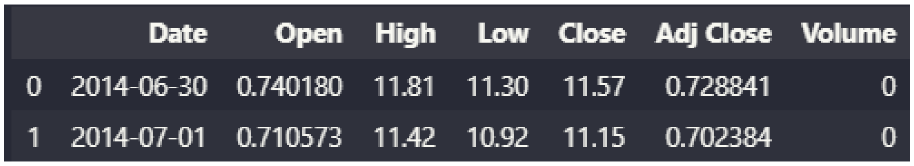
The MSE is a commonly used metric to evaluate the performance of regression models, including implied volatility models. This is because implied volatility is a continuous variable, and the MSE measures the average squared difference between the predicted and actual values. A lower MSE indicates better accuracy of the model's predictions.

In the context of implied volatility models, a low MSE indicates that the model is accurately predicting the implied volatility of an option based on its market price and other relevant factors, such as its demand and supply (5paisa Research Team, 2022). This is important for traders and investors who use implied volatility as an input in their decision-making process (Nickolas & Silberstein, 2022). By having a reliable and accurate model, they can make better-informed decisions about when to buy or sell options. While there are other available metrics that can be considered, MSE (Mean Squared Error) is a popular choice for evaluating regression models over MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) for several reasons (Deval, 2022):

- It emphasizes larger errors: Squaring the errors in MSE means that larger errors will be weighted more heavily than smaller ones. In option pricing, it is important to accurately capture extreme values of the option prices, such as during market crashes or large price movements, and MSE penalizes large deviations from the true value more heavily than mean absolute error (MAE).
- It has a unique global minimum: MSE is a smooth, continuous function that has a unique global minimum. This makes it easier to optimize using gradient-based techniques.
- It is more sensitive to outliers: The squared term in MSE means that outliers have a greater impact on the overall error than in MAE or RMSE.

Overall, using MSE as an evaluation metric in option pricing models can help to ensure that the model is accurately capturing extreme values and is better able to deal with large errors, which is essential for pricing options in real-world markets.

3.1.3 Analysis of data



	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-06-30	0.740180	11.81	11.30	11.57	0.728841	0
1	2014-07-01	0.710573	11.42	10.92	11.15	0.702384	0

Figure 3. VIX Dataset with headers

In order to understand the value of having more features in a machine learning model, I included the VIX Close data into the original Options Data as a feature. In theory, the VIX, which tracks the volatility of the S&P should improve our model performances.

On first joining the two datasets, with reference to Table 2, the correlation plot showed a low correlation with the implied volatility contrary to our intuition. One reason why the VIX may not be highly correlated to the implied volatility change of a given option in the S&P is that the VIX is based on the implied volatilities of a broad range of S&P 500 index options, rather than just one option. As such, the VIX may be influenced by changes in the implied volatilities of options with different maturities, strikes, and other characteristics than the specific option being analyzed.

Additionally, the VIX is calculated (Fidelity International, n.d.) using a complex methodology that takes into account a variety of factors, including the bid-ask spread of each option and the time to expiration of each option that may introduce noise and other distortions that can obscure a relationship between the VIX and the implied volatility of a specific option. Finally, it is possible that other factors, such as changes in interest rates, market sentiment, or macroeconomic indicators, may also impact the VIX and the implied volatility of specific options in different ways, further reducing the correlation between the two.

3.1.3 Analysis of results

Our results showed that the neural network model significantly outperformed the logistic regression model in predicting the implied volatility of options data. Specifically, the neural network model achieved lower MSE values on the testing set without additional features. This shows the potential that neural networks have when analysing implied volatility models, with an overall improvement in performance by 15.71%.

Linear Regression Test loss without VIX (MSE)	7.632364938220805e-05
Neural Network Test Loss without VIX (MSE)	6.43347084405832e-05
Gain Ratio	15.71 %

Table 3: MSE results of linear regression model against NN model

Furthermore, incorporating the VIX as a feature greatly improved the performance of both models, thereby emphasizing the importance of considering the market's volatility when predicting the implied volatility of options, with a new gain ratio of 30.54%. In this case, we see that simply adding one feature can improve the performance of a model by almost 2 times.

Linear Regression Test loss with VIX (MSE)	7.552458231722083e-05
Neural Network Test Loss with VIX (MSE)	5.24568313267082e-05
Gain Ratio with VIX	30.54 %

Table 3: MSE results of linear regression model against NN model

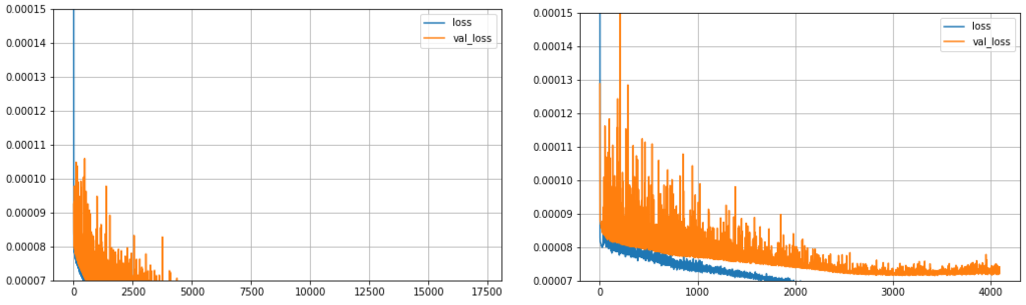


Figure 4. Analysis without VIX (left) and with VIX (right)

Furthermore, in terms of time-complexity, another feature that many companies consider (Analysts are salaried by the hour) we can see that the plot above in epochs for an untrained model, the addition of VIX as a feature has allowed the model to complete its training in almost 1000 less epochs on validation sets, resulting in a far fewer number of hours needed to compile the findings with more features.

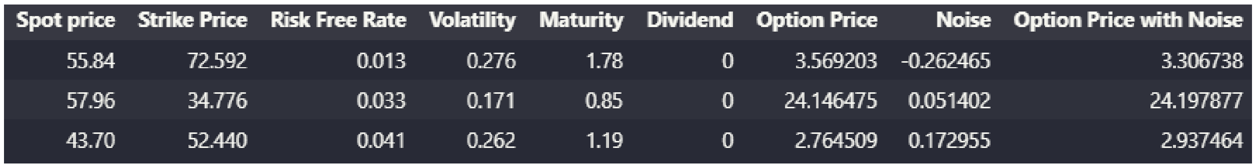
In conclusion, our study showcases the efficacy of using neural networks for predicting the implied volatility of options data, as well as the importance of incorporating relevant features such as the VIX. Our findings are particularly significant, as they could have far-reaching implications in the field of financial analytics. By showcasing a deeper understanding of the underlying complexities, our research provides valuable insights into this important aspect of financial markets.

3.2. Hyperparameter tuning on option prices with neural networks

3.2.1 Exploratory data analysis and pre-processing

The pricing of options is a complex and essential aspect of financial analytics, and neural networks have shown tremendous potential in improving the accuracy of option pricing models. In this study, I aim to improve the accuracy of option pricing models by comparing the famous Black Scholes Merton model (BSM) against neural network, demonstrating this in conjunction with tuning of the hyperparameters in neural networks. For this first half, I will use Hull's original Options Dataset, as the data is clean and reliable.

In the later half of this analysis, I will perform a similar comparison between the BSM against the neural network model using cryptocurrency options in the algorithmic portion of the hyperparameter tuning, to see if traditional financial models are applicable to newer non-traditional markets.



Spot price	Strike Price	Risk Free Rate	Volatility	Maturity	Dividend	Option Price	Noise	Option Price with Noise
55.84	72.592	0.013	0.276	1.78	0	3.569203	-0.262465	3.306738
57.96	34.776	0.033	0.171	0.85	0	24.146475	0.051402	24.197877
43.70	52.440	0.041	0.262	1.19	0	2.764509	0.172955	2.937464

Figure 5. Option prices data

For the clean options dataset, as the data does not have null values, we can perform scaling in preparation for fitting the sequential models in the neural network. Z-score scaling (Google Developers, n.d.) is a technique used to normalize data by subtracting the mean of the data and dividing it by the standard deviation. This technique is beneficial for option price data because it allows for the comparison of prices across different options with varying strike prices and expiration dates.

Option prices can vary widely depending on the underlying asset, time to expiration, and strike price. By using Z-score scaling, we can standardize the price data and express it in terms of the number of standard deviations away from the mean. This normalization makes it easier to compare the prices of different options and identify patterns or anomalies in the data.

3.2.2 Performance measures

In my particular analysis, I will use the test MAE (Mean Absolute Error), Mean Error, and Standard Error to evaluate the accuracy of neural network models in option pricing.

Test MAE (Acharya, 2021) is a measure of the average absolute difference between the predicted option prices and the actual option prices in the test set. This metric is beneficial for option pricing because it provides a straightforward and interpretable measure of the average magnitude of the errors made by the neural network model. Additionally, MAE is less sensitive to outliers than other metrics like Mean Squared Error (MSE), making it more suitable for option pricing data that may contain extreme values.

Mean Error (Statistics How To, n.d.) is a measure of the average difference between the predicted option prices and the actual option prices in the test set. This metric provides information on the direction and magnitude of the errors made by the model. Positive values indicate that the model overestimates the option prices, while negative values indicate that the model underestimates the option prices. Mean Error is useful for option pricing because it can provide insights into the bias of the model.

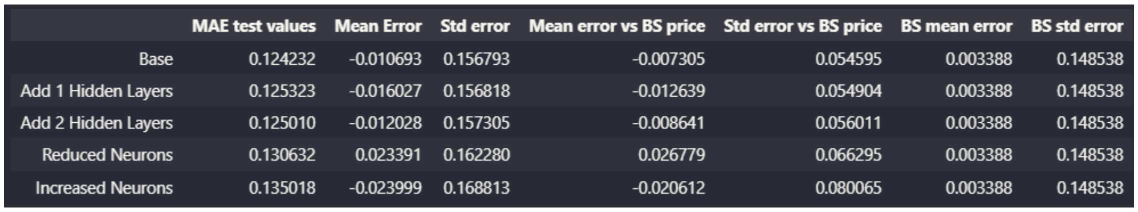
Standard Error (Kenton, 2022) is a measure of the variability in the errors made by the neural network model. It can help determine whether the model is consistently making errors or if the errors are randomly distributed. Standard Error is particularly useful for option pricing because it can provide insights into the level of uncertainty associated with the predictions made by the model.

Overall, Test MAE, Mean Error, and Standard Error are useful performance measures for option pricing with neural networks because they provide interpretable and meaningful measures of the accuracy, bias, and variability of the model's predictions.

3.2.3 Analysis of results

3.2.3.1 Manual tuning

Our manual tuning process involves manually tuning each sequential model in Keras by comparing several models: the base layer, adding 1 layer, adding 2 layers, reducing the number of neurons, and then increasing the number of neurons. The base layer contains 2 hidden layer, 1 input layer and 1 output layer each with 20 neurons, with the "sigmoid" activation function.



	MAE test values	Mean Error	Std error	Mean error vs BS price	Std error vs BS price	BS mean error	BS std error
Base	0.124232	-0.010693	0.156793	-0.007305	0.054595	0.003388	0.148538
Add 1 Hidden Layers	0.125323	-0.016027	0.156818	-0.012639	0.054904	0.003388	0.148538
Add 2 Hidden Layers	0.125010	-0.012028	0.157305	-0.008641	0.056011	0.003388	0.148538
Reduced Neurons	0.130632	0.023391	0.162280	0.026779	0.066295	0.003388	0.148538
Increased Neurons	0.135018	-0.023999	0.168813	-0.020612	0.080065	0.003388	0.148538

Figure 6. Results obtained from the models

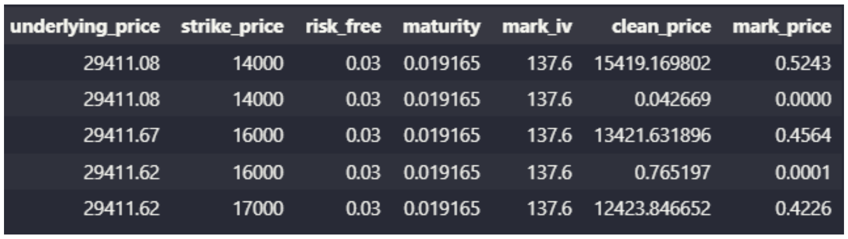
To assess the performance of my models, I compared the predicted option prices to the actual option prices obtained from the Black-Scholes (BS) model. I observed that the mean error was higher for the BS model compared to all neural networks, while the standard error of the neural networks was smaller than that of the BS model. My first conclusion that can be drawn from this is that the neural networks outperformed the BS model in terms of predicting option prices, especially in the mean error.

Regarding the manual tuning process, I observed that the mean absolute error was minimized for the base model, while the mean error was smallest for the 2 hidden layer model, and the standard error was smallest for the 1 hidden layer model. This suggests that the number of layers and neurons in the network can impact the model's performance and should be carefully chosen.

3.2.3.1 Algorithmic tuning

For the next part of our study, I utilized Deribits API to download the most updated option price data, in order to test the adequacy of the BSM in a non-traditional market context.

I then preprocessed the data, including cleaning the data and creating the time to maturity column. Next, I used the BSM formula to calculate the clean price of each of the options based on the available features in the dataset.



underlying_price	strike_price	risk_free	maturity	mark_iv	clean_price	mark_price
29411.08	14000	0.03	0.019165	137.6	15419.169802	0.5243
29411.08	14000	0.03	0.019165	137.6	0.042669	0.0000
29411.67	16000	0.03	0.019165	137.6	13421.631896	0.4564
29411.62	16000	0.03	0.019165	137.6	0.765197	0.0001
29411.62	17000	0.03	0.019165	137.6	12423.846652	0.4226

Figure 7. BTC Options data with clean_price and maturity as new features

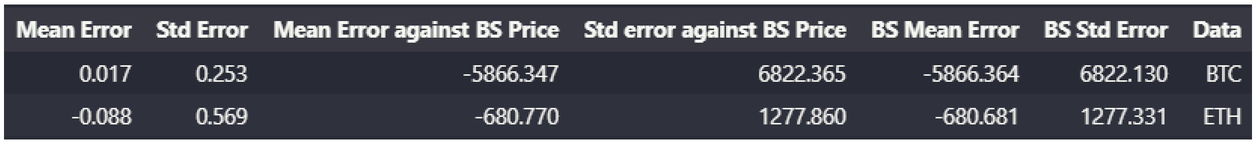
I observed that the pricing of BTC and ETH options differed significantly due to differences in how these options are priced. Specifically, BTC options are priced using the American model, while ETH options are priced using the European model. These differences (Majaski & Silberstein, 2022) in pricing led to significant differences in the prices predicted by the BS model.

To tune the model, I employed the Keras tuner to find the best hyperparameters and optimal epochs, with minimizing the mean absolute error as the goal for the algorithm. The results can be seen in table 4.

NN Model	Optimal number of units in first dense layer	Optimal learning rate	Best Epoch (MAE)
BTC	352	0.0001	46764
ETH	128	0.0001	49131

Table 4. Optimal Hyperparameters

Unfortunately, the neural network was unable to effectively predict the prices of cryptocurrency derivatives, especially in terms of standard error, with a far higher standard error than when used to price traditional options. However, we can see that the mean error and standard errors are far greater than that of the NN models, showing that traditional financial models may not be as useful for pricing crypto options.



Mean Error	Std Error	Mean Error against BS Price	Std error against BS Price	BS Mean Error	BS Std Error	Data
0.017	0.253	-5866.347	6822.365	-5866.364	6822.130	BTC
-0.088	0.569	-680.770	1277.860	-680.681	1277.331	ETH

Figure 8. Performance measures for Hyperparameter tuned NN model on BTC & ETH

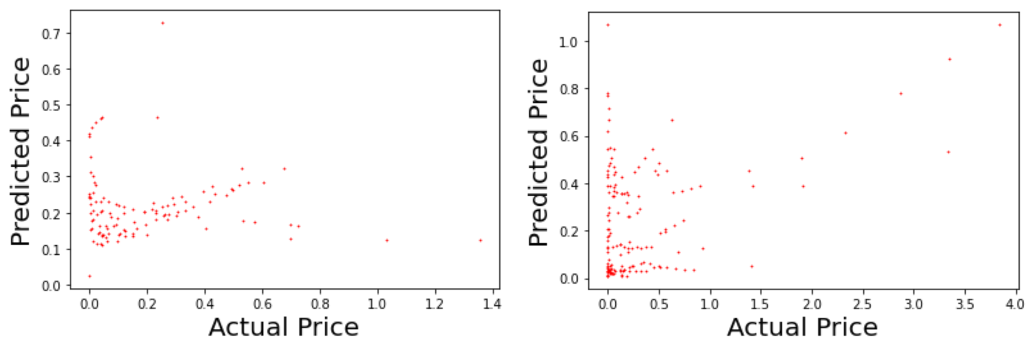


Figure 9. Performance of NN models BTC (left) & ETH (right) on the test set

In conclusion, our study demonstrates the potential of neural networks for improving option pricing accuracy through manual tuning, as well as through algorithmic tuning. Doing so, we highlight the importance of carefully choosing the number of layers and neurons in the network and the impact of differences in pricing models on option pricing accuracy. However, our findings also suggest that the effectiveness of neural networks in predicting the prices of cryptocurrency derivatives is still limited, emphasizing the need for further research in this area.

3.3 Credit lending problem with neural networks

Neural networks have become an increasingly popular tool in the field of credit lending, thanks to their ability to analyze complex data sets and make accurate predictions. With the rise of machine learning and big data, financial institutions are turning to these powerful algorithms to improve their credit scoring models and better assess the risk of lending to borrowers. Neural networks have the capacity to process vast amounts of data, recognize patterns, and make predictions with high accuracy. By leveraging these capabilities, banks and other lenders can make more informed lending decisions, leading to reduced risk and improved profitability. In this way, neural networks are proving to be an invaluable tool in solving the credit lending problem. In this pillar, I will explore the gigantic credit lending dataset with over 2 million data points, applying big data techniques for cleaning large scaled data, before using the algorithmic hypertuning method from the second pillar to tune my neural network model.

3.3.1 Exploratory data analysis and pre-processing

3.3.1.1 Feature engineering

I conducted a thorough exploratory data analysis (EDA) and pre-processing of data to fit various machine learning classification models. In considering the features at a glance, I computed FICO_avg, payment_income_ratio, delinq_2yrs, inq_last6mths, and pub_rec, features that are commonly used in credit risk analysis, and are based on a person's credit history, income, and other financial indicators (Stobierski, 2020).

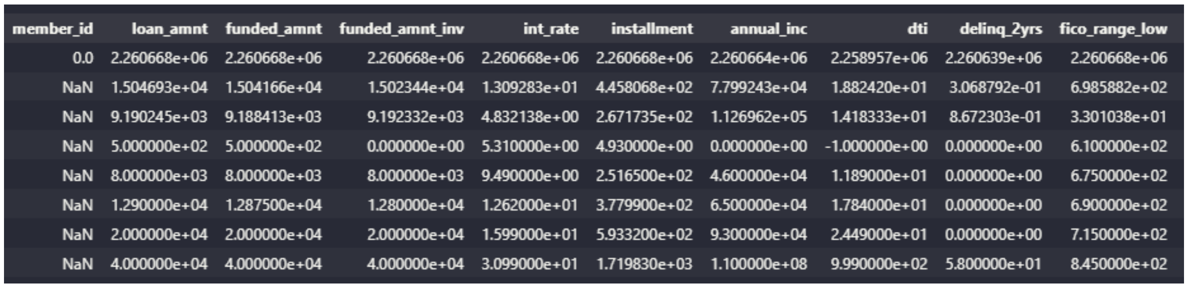
Next, I grouped open_acc, annual_inc, and dti into bins to easily categorize the features. Grouping these variables into bins makes it easier to interpret and compare the effects of different values on the outcome variable. For example, if I group annual income into different income brackets, I can see if people in higher income brackets are less likely to default on loans than those in lower income brackets.

3.3.1.2 Big data techniques

When dealing with big data and large datasets, we want to perform many complex operations at the same time on similar values rather than on an individual level (i.e iteratively). As a first step, I removed columns that contained over 25% NA or null values. This is a standard practice in data cleaning as columns with a large number of missing values can't provide much useful information. Next, I removed columns with too many unique strings such as id, emp_title, and title, which had over 2m unique values which likely contain irrelevant information that will not contribute much to the predictions.

In dealing with string format data, I created an algorithm to extract numbers, for example, if the loan period was 24_months, 24 would be saved as an integer, ensuring care in dropping the first item to prevent the dummy variable trap. This process quickly converts text data into numeric data, which can be used in machine learning models.

I then converted all NA values to 0 for the sake of the other machine learning models, with an alternative being to impute the missing data. However, imputing values can distort the distribution of the data: Imputing missing values can change the distribution of the data, which may affect the accuracy of your model. If the missing values are not missing at random and are related to the target variable, imputing values may lead to misleading predictions. Furthermore, neural networks can handle missing data: Neural networks can handle missing data by using techniques such as dropout and regularization, which can improve the performance of the model without imputing missing values.



member_id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	delinq_2yrs	fico_range_low
0.0	2.260668e+06	2.260668e+06	2.260668e+06	2.260668e+06	2.260668e+06	2.260664e+06	2.258957e+06	2.260639e+06	2.260668e+06
NaN	1.504693e+04	1.504166e+04	1.502344e+04	1.309283e+01	4.458068e+02	7.799243e+04	1.882420e+01	3.068792e-01	6.985882e+02
NaN	9.190245e+03	9.188413e+03	9.192332e+03	4.832138e+00	2.671735e+02	1.126962e+05	1.418333e+01	8.672303e-01	3.301038e+01
NaN	5.000000e+02	5.000000e+02	0.000000e+00	5.310000e+00	4.930000e+00	0.000000e+00	-1.000000e+00	0.000000e+00	6.100000e+02
NaN	8.000000e+03	8.000000e+03	8.000000e+03	9.490000e+00	2.516500e+02	4.600000e+04	1.189000e+01	0.000000e+00	6.750000e+02
NaN	1.290000e+04	1.287500e+04	1.280000e+04	1.262000e+01	3.779900e+02	6.500000e+04	1.784000e+01	0.000000e+00	6.900000e+02
NaN	2.000000e+04	2.000000e+04	2.000000e+04	1.599000e+01	5.933200e+02	9.300000e+04	2.449000e+01	0.000000e+00	7.150000e+02
NaN	4.000000e+04	4.000000e+04	4.000000e+04	3.099000e+01	1.719830e+03	1.100000e+08	9.990000e+02	5.800000e+01	8.450000e+02

Figure 10. Snippet of lending club data after processing

Finally, I converted the remaining categorical fields to dummies using the pd.getdummies() function in a process called one-hot encoding and is used to represent categorical variables as binary vectors. This allows the machine learning models to understand the categorical variables and improves their performance. Overall, by performing these techniques, the dataset of over 2 million points can be converted to an entirely float and integer based dataframe that can be used for both regression and classification predictions.

3.3.2 Performance measures

I then explored the number of accepted loans to the rejected loans and found that only 1.2% of loans default, which means that the data is highly imbalanced. This is a common problem in credit risk analysis, as the vast majority of loans are repaid on time, making it challenging to accurately predict default risk. Based on this, I prioritized AUC Score, F1 scores, and precision scores for my analysis. These metrics are better suited to imbalanced datasets as they take into account both true positives and false positives.

3.3.3 Model selection and performance

I used five different models: logistic regression, random forest classifier, XGBoost classifier, an ensemble with the three previous models, and a neural network model using the findings from the second pillar to compare the performance of each model on predicting.

In selecting the models for this analysis, I chose different models based on their properties in conjunction with our dataset. Below details the reason for selecting each of the models:

1. Logistic regression was chosen as it is simple to interpret and fast to train (Hale, 2019). However, it assumes the independence of the input variables and the linearity of the relationship between the input variables and the output. These assumptions may not hold in real-world contexts, leading to potential performance compromise.
2. Random forest is a tree-based ensemble model that uses decision trees. I chose this model for its ability to make quick predictions (Yiu, 2019). However, it can be computationally expensive and may not perform well on highly imbalanced datasets such as the lending club dataset.
3. XGBoost is a gradient boosting model that ensembles weak learners, making fewer assumptions and requiring fewer hyperparameters to tune (Jain, 2016). It was chosen for its high interpretability and fast training times (Lundberg, 2018), making it suitable for our large-scale lending club dataset. In contrast to the random forest, XGBoost performs well on highly imbalanced datasets and has been shown to outperform other classification models in various applications.
4. An ensemble model combining the logistic regression, random forest and XGBoost models was chosen as it combines multiple algorithms that may improve the accuracy of predictions (Brownlee, 2020). Each algorithm has its own strengths and weaknesses, and combining them can help to mitigate these weaknesses and provide more robust predictions. For instance, logistic regression can be useful for modeling linear relationships, while random forest can be useful for modeling nonlinear relationships, and XGBoost can be useful for modeling complex interactions between variables (Pardy, 2020).
5. The neural network model is a powerful tool for processing large scale datasets like the lending club data. Neural networks can learn complex patterns and relationships within the data that may be difficult to identify when using other machine learning algorithms. Additionally, neural networks can be designed to handle highly imbalanced data, which is often the case with lending club data, where the number of non-defaulting loans typically far outweighs the number of defaulting loans (Chen, 2023).

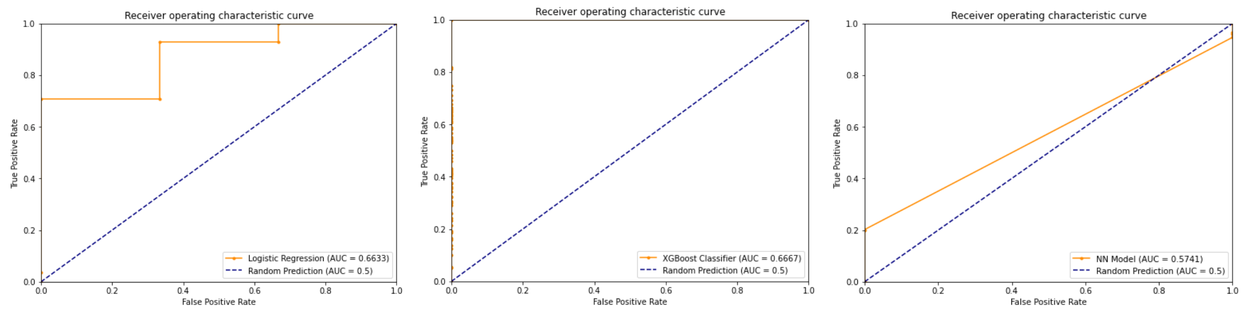


Figure 11. AUC-ROC for better performing models in terms of AUC (LR, XGB, Ensemble, NN)

On a small subset of the dataset, I plotted the AUC-ROC for each model and compared the performance measures of each model, determining that XGBoost was the best performing model by a narrow margin.

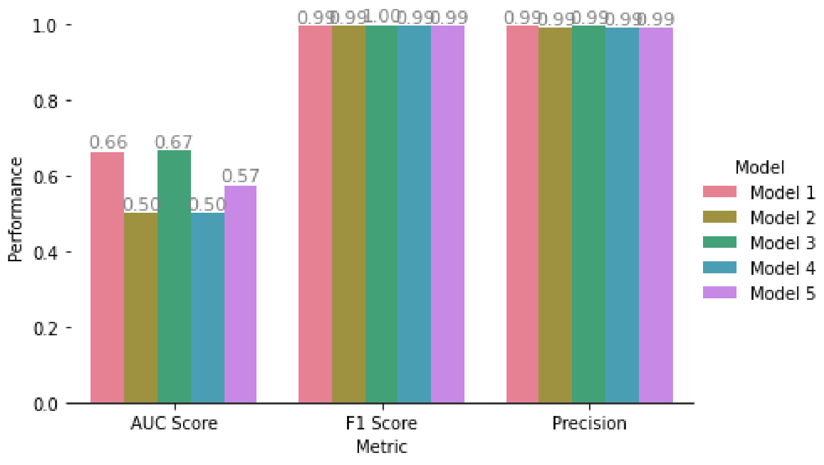
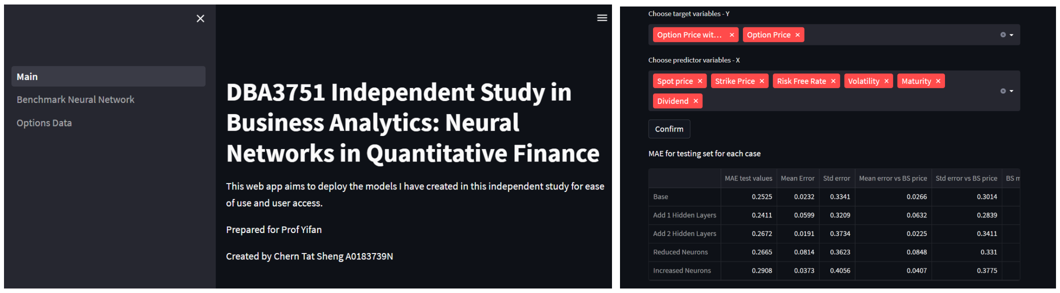


Figure 12. Performance of each model based on AUC, F1 and Precision
(From left to right: LR, RF, XGBoost, Ensemble, NN)

I chose XGBoost with parameters {'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 100} over the other models as it not only performed the best out of the models selected, but also time efficient, and makes less assumptions compared to the similarly performing logistic regression. Unfortunately, while the neural networks are powerful models, the complexity and time it took to fully tune the model made it difficult to justify the minor performance gain, given the same run time as the other models. However, if given a sufficient amount of epochs to tune and perhaps a larger amount of data points (as I only used a fraction to save time), it is likely that we can see an incrementally higher performance in the model.

4. Final Product

Using what I had learnt from the research pillars, I developed an app as a personal project to challenge myself and push my abilities to the limit. As someone who has always been interested in the field of data science and machine learning, I saw an opportunity to explore and experiment with building data pipelines, packaging it in a web application. (Appendix A:User Guide)



DBA3751 Independent Study in Business Analytics: Neural Networks in Quantitative Finance

This web app aims to deploy the models I have created in this independent study for ease of use and user access.

Prepared for Prof Yifan

Created by Chern Tat Sheng A0183739N

Choose target variables - Y: Option Price wit… ×, Option Price ×

Choose predictor variables - X: Spot price ×, Strike Price ×, Risk Free Rate ×, Volatility ×, Maturity ×, Dividend ×

MAE for testing set for each case

	MAE test values	Mean Error	Std error	Mean error vs BS price	Std error vs BS price	BS m
Base	0.2525	0.0232	0.3341	0.0266	0.3014	
Add 1 Hidden Layers	0.2411	0.0599	0.3209	0.0632	0.2839	
Add 2 Hidden Layers	0.2672	0.0191	0.3734	0.0225	0.3411	
Reduced Neurons	0.2665	0.0814	0.3623	0.0848	0.331	
Increased Neurons	0.2908	0.0373	0.4056	0.0407	0.3775	

Figure 13. Pages and demo for the web application

The first part of the app that I developed was the neural network tuner for options, which accepts a CSV file with any number of columns. The user is then asked to select the target and predictor features and several neural network models are fitted and compared to return the results in a tabular format. Since the neural network tuner was derived from pillar 2 of my research project, which compared the Black Scholes model against the neural network, it required the option price with noise as well as the clean option price to fit and test the models. Upon running these models, the neural network tuner will return the results for mean error and standard error for both the Black Scholes model and the neural network model.

In the recent weeks, I applied my findings from the previous weeks and built the second part of the app as a classifier designed to handle large amounts of data, such as the Lending Club data. The application accepts any CSV file, and the user is allowed to enter a target variable and manually group all values into two bins. The app performs a similar big data cleaning as in Section 3.3.1.2.

As a test case, I chose a snippet of the large lending club dataset. After uploading the dataset, the user will select the loan_status column and be allowed to create their own class based on what they consider as a default/non-default. To develop the classifier, I drew inspiration from pillar 3 of my research project, where I had learned about different types of classification models. Finally, the user can select from a range of five different models to run the model on and submit the query to obtain the AUC score, F1 score and precision metrics.

In conclusion, developing this app was a significant accomplishment for me, and it allowed me to challenge myself and expand my knowledge and skill set in the field of data science and machine learning in quantitative finance.

5. Conclusion

In conclusion, my experience in building the app and conducting research on neural networks and machine learning has been a valuable learning experience. Through this process, I have gained several key insights:

- The usefulness of neural network pipelines: The implementation of neural network pipelines in my app proved to be an efficient and effective way to package machine learning functions and make them more accessible to end-users.

- The importance of selecting appropriate evaluation metrics: The research I have performed in the last weeks, applied onto the data, shows that different metrics can be more appropriate for different applications, and it's important to carefully consider the specific goals of a project when selecting evaluation metrics.
- The value of continuing to learn and research: By staying informed about the latest trends and techniques, we can continue to improve the accuracy and effectiveness of our models. For example, Keras tuner was refined greatly and only rose in popularity after Dec 2022 (Not findable with ChatGPT!).

Finally, our research has shown the tremendous potential for machine learning to make a positive impact in a wide range of fields, especially in quantitative finance. Thank you for your time in making it to the end of my report!

References

Acharya, S. (2021, May 14). *What are RMSE and MAE?* Towards Data Science. Retrieved April 21, 2023, from https://towardsdatascience.com/what-are-rmse-and-mae-e405ce230383

Brownlee, J. (2020, October 26). *Why Use Ensemble Learning? - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved April 21, 2023, from https://machinelearningmastery.com/why-use-ensemble-learning/

Chen, J. (2023, February 28). Default: What It Means, What Happens When You Default, Examples. Retrieved April 21, 2023, from https://www.investopedia.com/terms/d/default2.asp

Deval, S. (2022, November 18). *Mean Squared Error: Definition, Applications and Examples*. Great Learning. Retrieved April 21, 2023, from https://www.mygreatlearning.com/blog/mean-square-error-explained/

Fidelity International. (n.d.). *VIX: What you should know about the volatility index*. Fidelity Singapore. Retrieved April 21, 2023, from https://www.fidelity.com.sg/beginners/what-is-volatility/volatility-index

5paisa Research Team. (2022, November 28). *What Is Implied Volatility & Factors Affecting It? | 5paisa*. 5Paisa. Retrieved April 21, 2023, from https://www.5paisa.com/stock-market-guide/derivatives-trading-basics/what-is-implied-volatility

Google Developers. (n.d.). *Normalization | Machine Learning*. Google Developers. Retrieved April 21, 2023, from https://developers.google.com/machine-learning/data-prep/transform/normalization

Hale, J. (2019, September 27). *Don't Sweat the Solver Stuff. Tips for Better Logistic Regression… | by Jeff Hale*. Towards Data Science. Retrieved April 21, 2023, from https://towardsdatascience.com/dont-sweat-the-solver-stuff-aea7cddc3451

Jain, A. (2016, March 1). *XGBoost Parameters Tuning | Complete Guide With Python Codes*. Analytics Vidhya. Retrieved April 21, 2023, from https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

Joseph, R. (2022, April 4). *Optimal ratio for data splitting*. Wiley Online Library. Retrieved April 21, 2023, from https://onlinelibrary.wiley.com/doi/full/10.1002/sam.11583

Kenton, W. (2022, September 6). *Standard Error (SE) Definition: Standard Deviation in Statistics Explained*. Investopedia. Retrieved April 21, 2023, from https://www.investopedia.com/terms/s/standard-error.asp

Kenton, W. (2022, September 18). *Hull-White Model*. Investopedia. Retrieved April 21, 2023, from https://www.investopedia.com/terms/h/hullwhite-model.asp

Lundberg, S. (2018, April 17). *Interpretable Machine Learning with XGBoost | by Scott Lundberg*. Towards Data Science. Retrieved April 21, 2023, from https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27

Majaski, C., & Silberstein, S. (2022, July 12). *American vs. European Options: What's the Difference?* Investopedia. Retrieved April 21, 2023, from https://www.investopedia.com/articles/optioninvestor/08/american-european-options.asp

Nickolas, S., & Silberstein, S. (2022, May 20). *Implied Volatility*. Investopedia. Retrieved April 21, 2023, from https://www.investopedia.com/ask/answers/032515/what-options-implied-volatility-and-how-it-calculated.asp

Pardy, S. (2020, May 12). *The XGBoost Model: How to Control It*. Capital One. Retrieved April 21, 2023, from https://www.capitalone.com/tech/machine-learning/how-to-control-your-xgboost-model/

Roshan, J. (2022, April 4). *Optimal ratio for data splitting*. Wiley Online Library. Retrieved April 21, 2023, from https://onlinelibrary.wiley.com/doi/full/10.1002/sam.11583

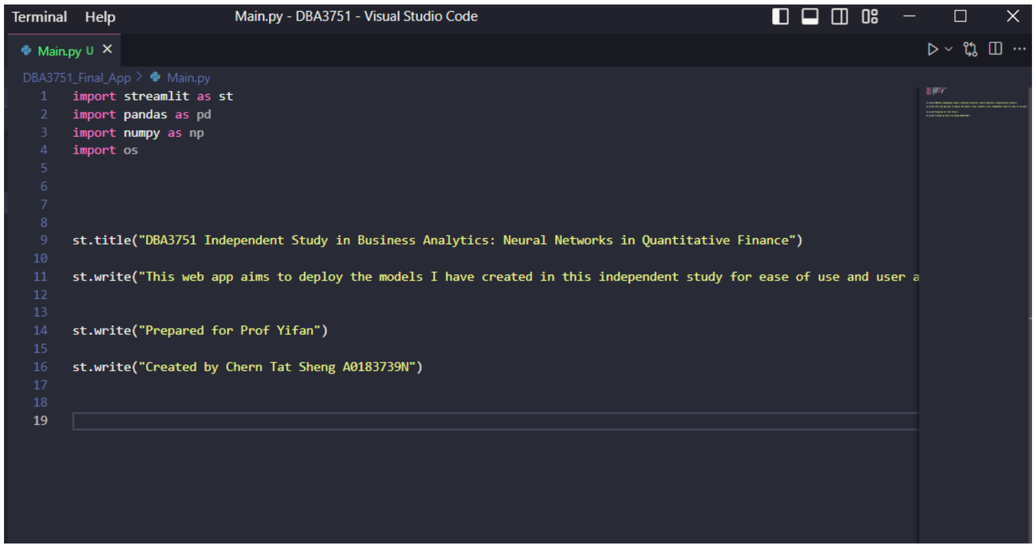
Statistics How To. (n.d.). *Mean Error*. Statistics How To. Retrieved April 21, 2023, from https://www.statisticshowto.com/mean-error/

Stobierski, T. (2020, May 5). *13 Financial Performance Measures Managers Should Monitor | HBS Online*. HBS Online. Retrieved April 21, 2023, from https://online.hbs.edu/blog/post/financial-performance-measures

Yiu, T. (2019, June 12). *Understanding Random Forest. How the Algorithm Works and Why it Is… | by Tony Yiu*. Towards Data Science. Retrieved April 21, 2023, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

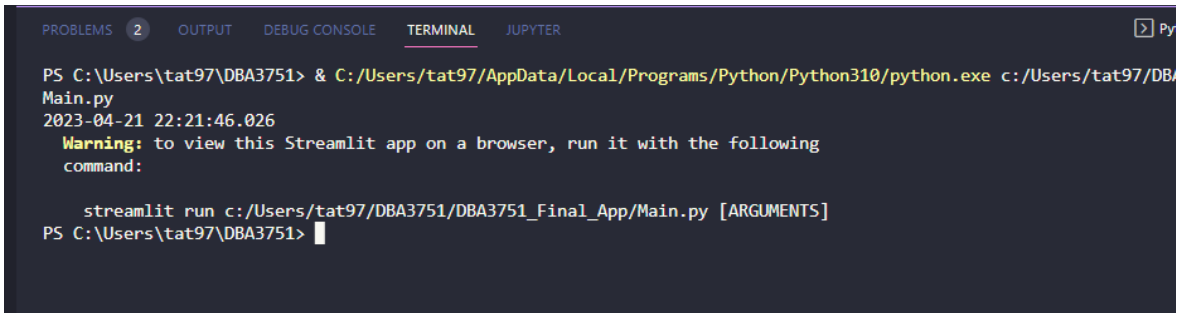
Appendix: Application User Guide

Step 1: Run the Main.py file, ensuring that it is your current working directory


Terminal
Help
Main.py - DBA3751 - Visual Studio Code
Main.py U
DBA3751_Final_App > Main.py

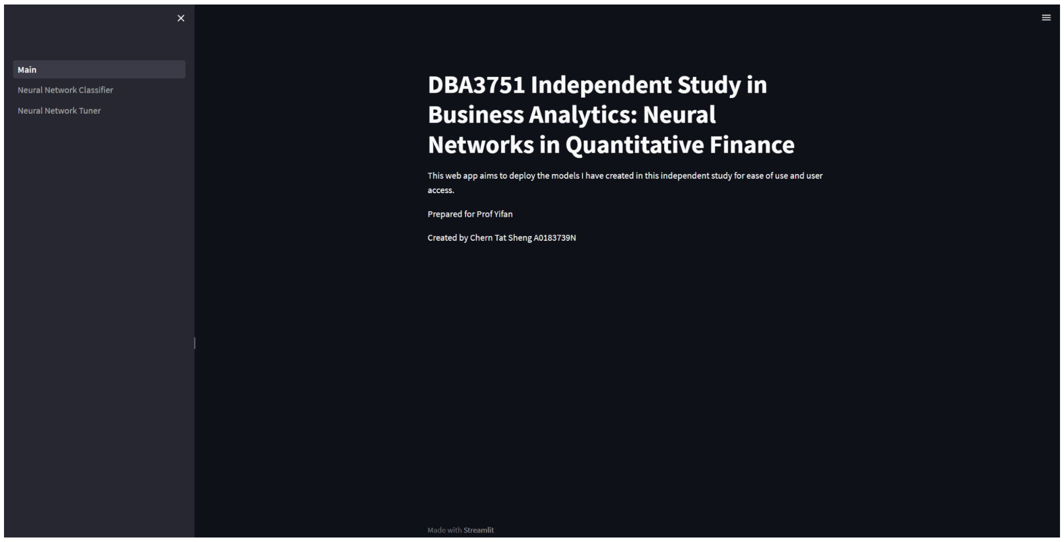
```python
import streamlit as st
import pandas as pd
import numpy as np
import os




st.title("DBA3751 Independent Study in Business Analytics: Neural Networks in Quantitative Finance")

st.write("This web app aims to deploy the models I have created in this independent study for ease of use and user a


st.write("Prepared for Prof Yifan")

st.write("Created by Chern Tat Sheng A0183739N")


```

Step 2: In the terminal, copy the command and run it again in the terminal.



```
PS C:\Users\tat97\DBA3751> & C:/Users/tat97/AppData/Local/Programs/Python/Python310/python.exe c:/Users/tat97/DB
Main.py
2023-04-21 22:21:46.026
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run c:/Users/tat97/DBA3751/DBA3751_Final_App/Main.py [ARGUMENTS]
PS C:\Users\tat97\DBA3751> 
```

Step 3: A popup will appear in your web browser containing the homescreen of the app.



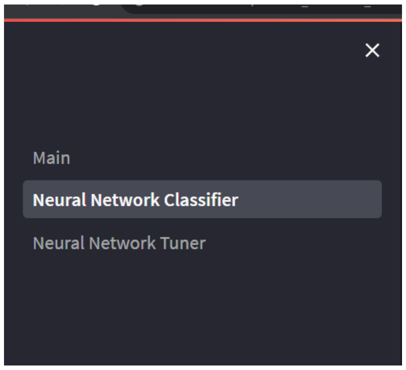
DBA3751 Independent Study in Business Analytics: Neural Networks in Quantitative Finance

This web app aims to deploy the models I have created in this independent study for ease of use and user access.

Prepared for Prof Yifan

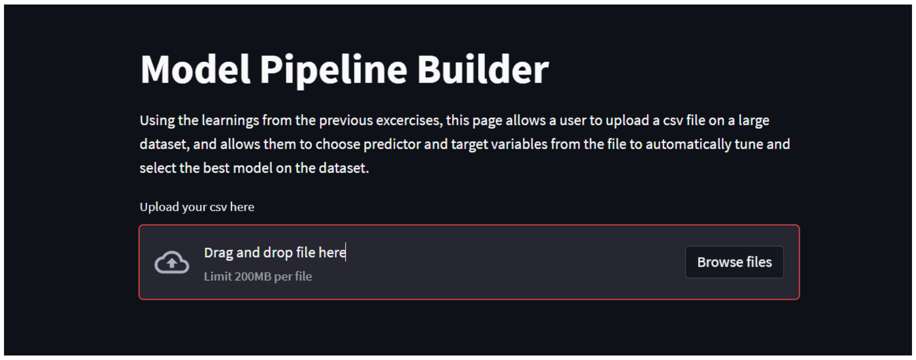
Created by Chern Tat Sheng A0183739N

Step 4: On the top left hand corner, click the tabs containing each part of the app.



Step 5: Upload the test files into the app

- Neural Network Classifier: Model Pipeline builder => shortened_test.csv
- Neural Network Tuner: Options data model => Options_Data Sample.csv

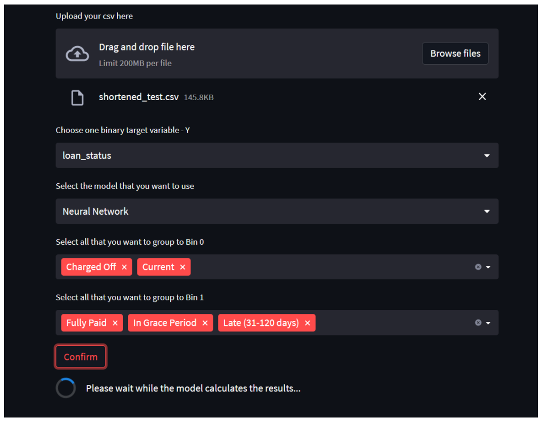


Model Pipeline Builder

Using the learnings from the previous excercises, this page allows a user to upload a csv file on a large dataset, and allows them to choose predictor and target variables from the file to automatically tune and select the best model on the dataset.

Upload your csv here

Step 6: Fill in the fields as per the instructions and confirm to run



Upload your csv here

shortened_test.csv 145.8KB

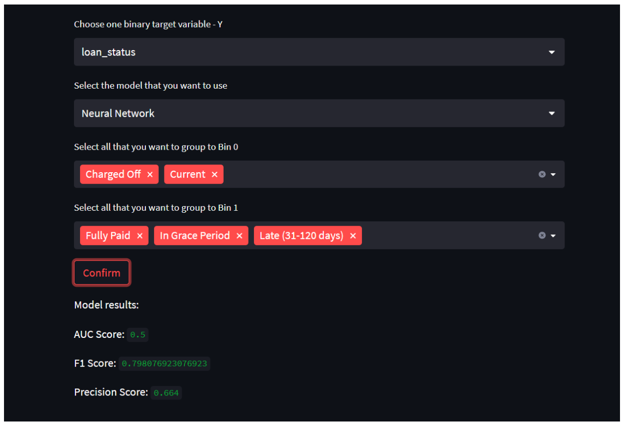
Choose one binary target variable - Y: loan_status

Select the model that you want to use: Neural Network

Select all that you want to group to Bin 0: Charged Off, Current

Select all that you want to group to Bin 1: Fully Paid, In Grace Period, Late (31-120 days)

Please wait while the model calculates the results...



Choose one binary target variable - Y: loan_status

Select the model that you want to use: Neural Network

Select all that you want to group to Bin 0: Charged Off, Current

Select all that you want to group to Bin 1: Fully Paid, In Grace Period, Late (31-120 days)

Model results:

AUC Score: 0.5

F1 Score: 0.798076923076923

Precision Score: 0.664