CREDIT DATA ANALYSIS AND MODELLING

1.0 OBJECTIVE

The following report has been completed to assess several different methods for classifying credit risk when applying for personal loans. The specific intent of this investigation is to determine which model is best for predicting whether an applicant will pose a potential repayment risk, based on a range of key criteria.

Analysis of this type will support the decision-making framework for businesses lending funds to individuals. The primary benefit being that the appropriate level of safeguards and lending restrictions can be applied on a case-by-case basis after the risk level is assessed using the chosen model.

2.0 DATA

The data set chosen contains data on 20 variables and the classification of whether an applicant is considered a Good or Bad credit risk for 1000 loan applicants. Many of these variables have been excluded from the initial data set load due to irrelevance for this analysis.

The remaining variables featured in this data set are:

Variable	Attribute
Age	Numeric
Sex	Text: male, female
Job	Numeric: 0 - unskilled and non-resident, 1 - unskilled and resident, 2 - skilled, 3 - highly skilled
Housing	text: own, rent, or free
Saving accounts	Text: little, moderate, quite rich, rich
Checking account	Numeric: currency
Credit amount	Numeric: currency
Duration	Numeric: in month
Purpose	Text: car, furniture/equipment, radio/TV, domestic appliances, repairs, education, business, vacation/others
Risk	Text: good or bad

- *Table 1 – features and attributes.*

Index	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	49	male	1	own	little	NaN	2096	12	education	good
3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	53	male	2	free	little	little	4870	24	car	bad

- *Table 2 – All columns in the dataset, with examples of the data types found.*

This data set will allow the classification of whether a future loan applicant is either a Good or Bad risk to the lender, depending on a combination of features that have historically been observed in a known customer base.

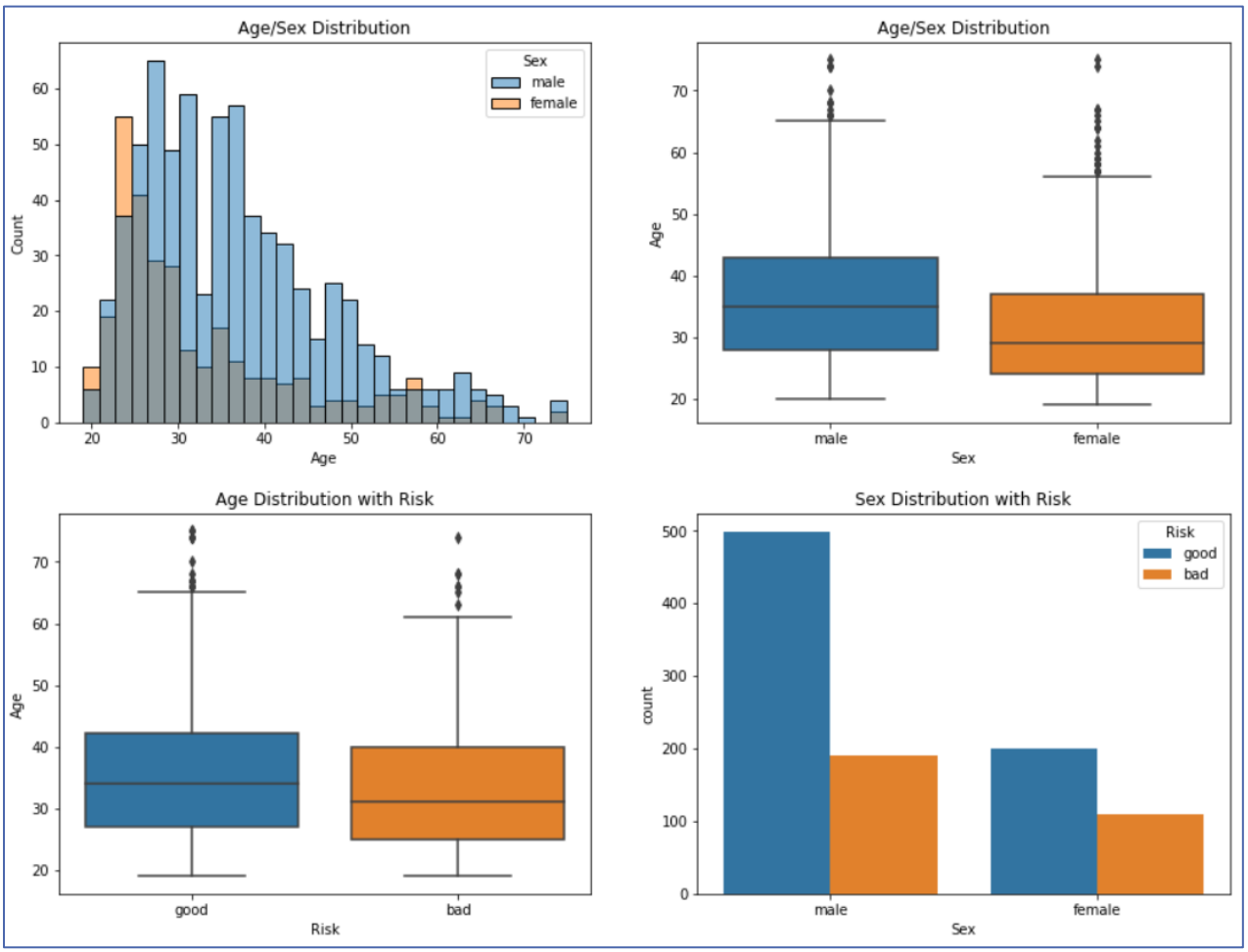
3.0 EXPLORATORY DATA ANALYSIS & FEATURE ENGINEERING

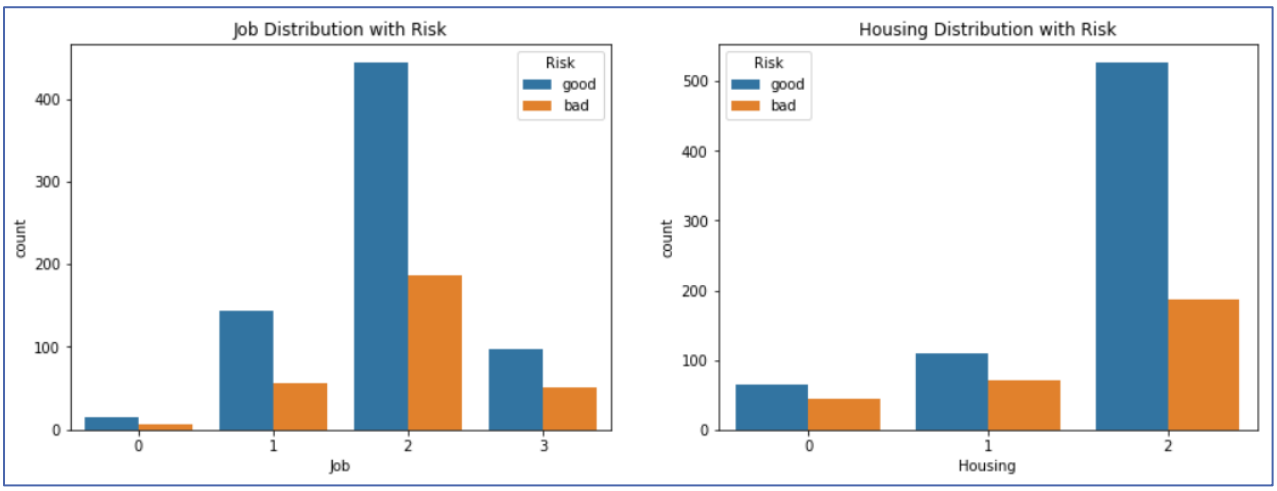
Analysis of the data was focused initially on determining which features are more likely to contribute to the result of either Good or Bad risk. Some feature engineering was required to normalise and standardise some of these features, depending on the regression model/s used.

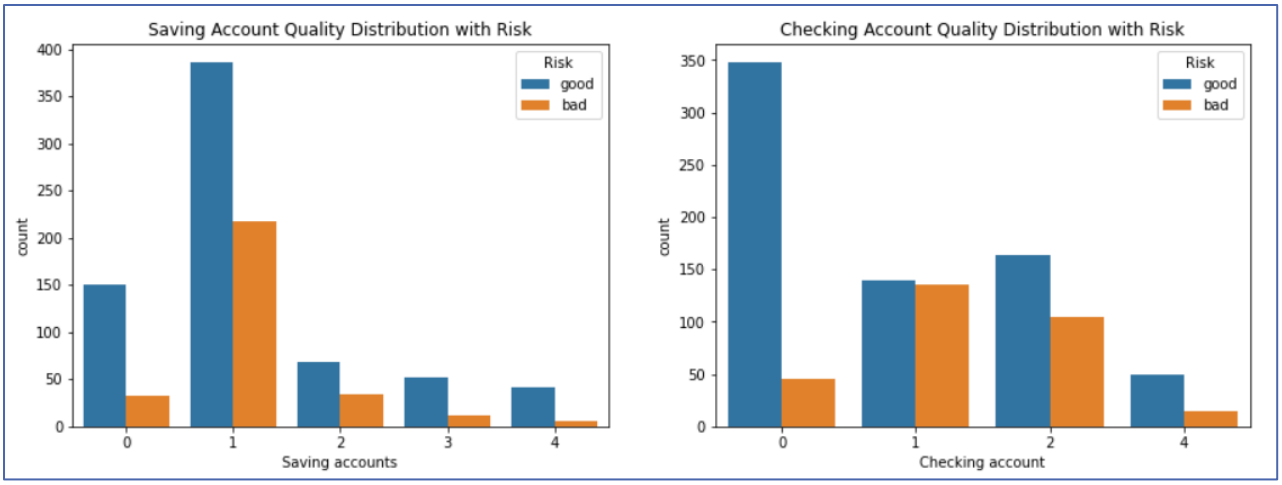
3.1 EDA

Several bar and box plots were created to see how these features compared against the associated risk.

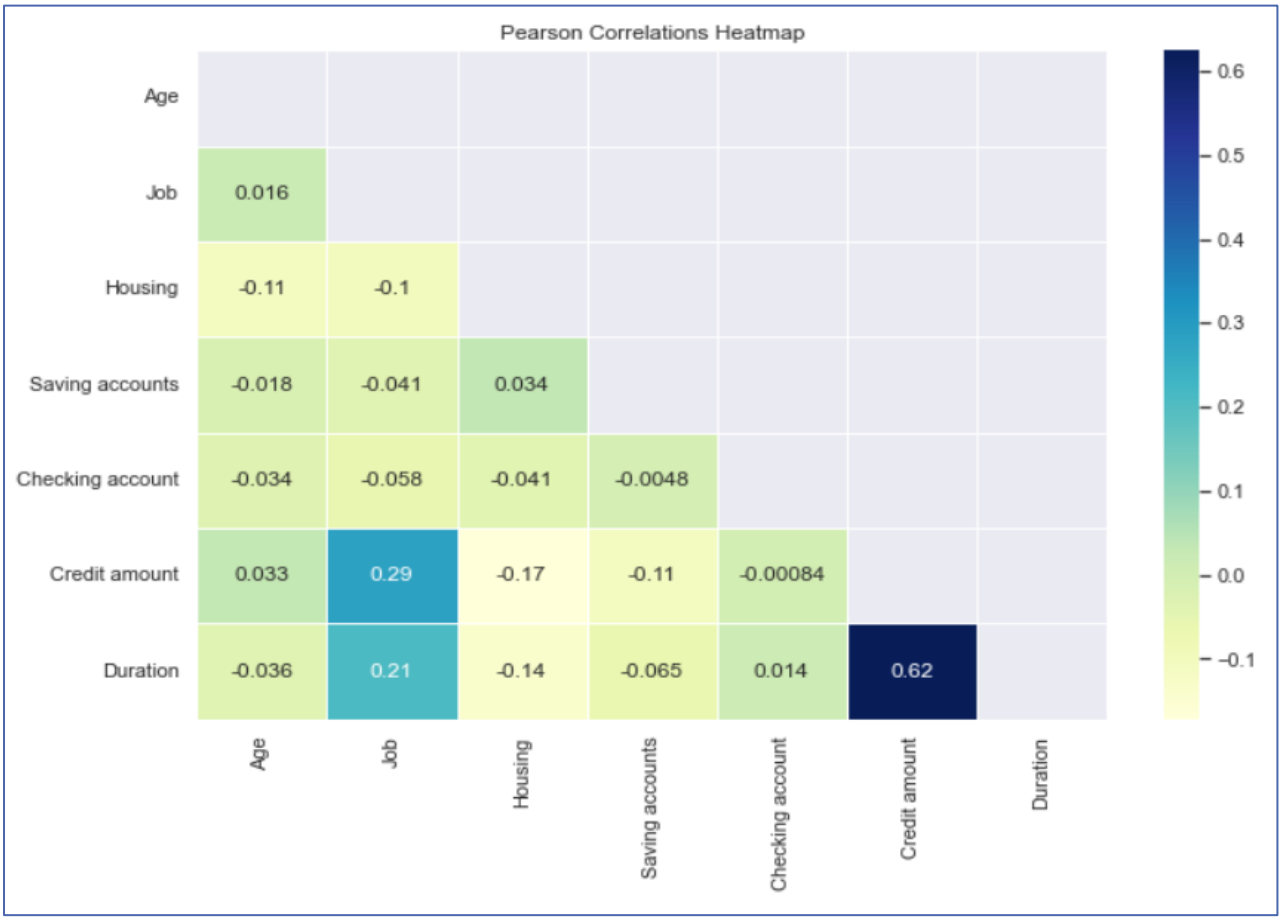
***Note that some feature engineering has already taken place prior to generating these visuals. Changes to the features will be explained in section 3.2.*







A PairPlot was also created using the following features: 'Age', 'Job', 'Housing', 'Saving accounts', 'Checking account', 'Credit amount', 'Duration', 'Risk'. However, the Pearson Correlation indicated much more clearly which features would be most likely to contribute to an outcome.



As a result, it was determined that only the following features would be considered for classification:

Feature
Credit Amount
Duration
Job
Savings Account
Checking Account

3.2 FEATURE ENGINEERING

The following steps were taken to prepare the data for use in the models.

- A new table was created by dropping the unwanted features 'Index','Age','Sex','Housing','Purpose'
- RISK was converted to a binary value (0 = Bad, 1 = Good)
- JOB is already categorised as a numeric value: 0 — unskilled and non-resident, 1 — unskilled and resident, 2 — skilled, 3 — highly skilled
- SC_LabelEncoder was applied to numerically categorise the SAVINGS and CHECKING account values

```
def SC_LabelEncoder(text):
    if text == "little":
        return 1
    elif text == "moderate":
        return 2
    elif text == "quite rich":
        return 3
    elif text == "rich":
        return 4
    else:
        return 0
```

- CREDIT AMOUNT and DURATION were left as is

The above actions provide a data set only containing values, which is far easier for our models to ingest and process than strings.

Finally, the remaining dataset was split into a train and test, with either 20% or 30% Test data, then scaling was applied before ingesting into the models.

4.0 CLASSIFIER MODELS

3 models were chosen for comparative analysis of this dataset.

4.1 LOGISTIC REGRESSION

Firstly, the modified dataset was run through a MinMax Scaler, then split into Training and Test sets with a 20% split.
Then a sklearn.linear_model.LogisticRegression model was defined with the following arguments:

```
penalty= 'l2'
multi_class = 'multinomial'
solver = 'lbfgs'
max_iter = 1000
```

Once the Logistic Regression model was fit, the results were viewed using an evaluate_metrics function which featured the below scores:

```
'accuracy': 0.685,
'recall': array([0.11666667, 0.92857143]),
'precision': array([0.41176471, 0.71038251]),
'f1score': array([0.18181818, 0.80495356])
```

In order to ensure that L2 was a good choice, a duplicate model was run using an L1 penalty. Here are the results:

```
'accuracy': 0.7,
'recall': array([0., 1.]),
'precision': array([0. , 0.7]),
'f1score': array([0.   , 0.82352941])
```

With the exception of precision, the L1 penalty method appears to be a slightly better fit.

4.2 K-NEAREST NEIGHBOUR

KNN is a simple classification method that is applicable in many real-life scenarios, as it does not make any underlying assumptions about the distribution of data. It will simply assign a category based on best fit amongst the specified number of nearby data points.

The model was fitted with scaled data (using StandardScaler), that was then split into train and test sets with a 30% test ratio.

The first test was based on n_neighbors=5. Evaluation of the performance of the classifier on test data using:

```
accuracy = knn.score(X_test_scaled, y_test)
print("Accuracy:", accuracy)
Accuracy: 0.7066666666666667
```

Further testing of this model using a wider range of n_neighbors, to confirm which will have the best accuracy score:

```
from sklearn.model_selection import cross_val_score

# Test different values of n_neighbors
k_values = [1, 3, 5, 7, 9]

for k in k_values:
knn = KNeighborsClassifier(n_neighbors=k)
scores = cross_val_score(knn, X_train_scaled, y_train, cv=5)
print("n_neighbors = {}: Accuracy = {:.2f} (+/- {:.2f})".format(k,
scores.mean(), scores.std() * 2))

n_neighbors = 1: Accuracy = 0.69 (+/- 0.09)
n_neighbors = 3: Accuracy = 0.71 (+/- 0.08)
n_neighbors = 5: Accuracy = 0.71 (+/- 0.07)
n_neighbors = 7: Accuracy = 0.73 (+/- 0.07)
n_neighbors = 9: Accuracy = 0.72 (+/- 0.08)
```

Therefore, based on the above it appears that the best value of ***k*** is 7, with an accuracy of 73%.

4.3 DECISION TREE CLASSIFIER

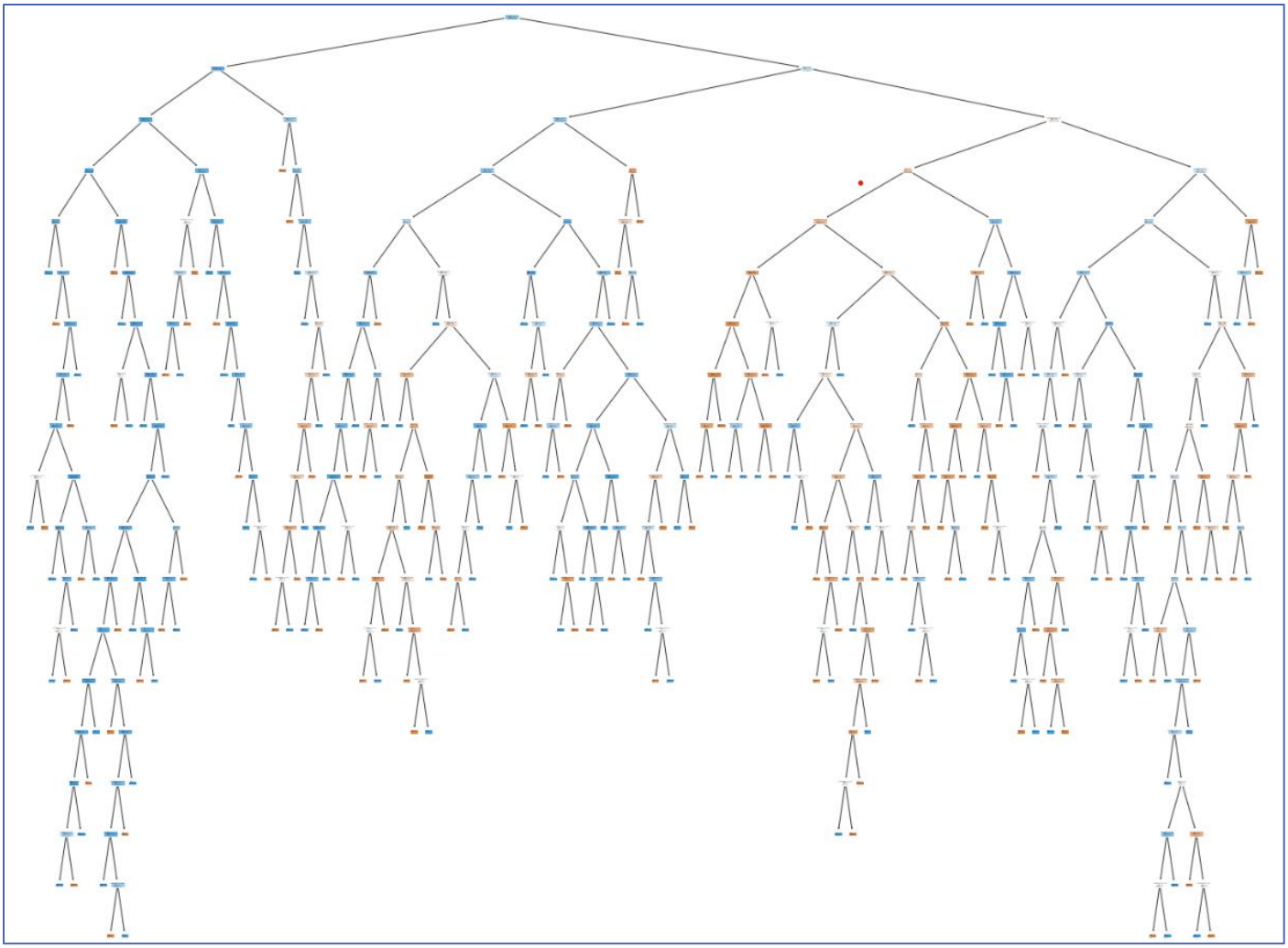
A Decision Tree is useful method of classification, where the result is typically of a binary nature (Yes or No, Good or Bad etc).

The data for the model was split into Train and Test sets, using a 20% split.

Evaluation of the model was completed using a similar method as in the Logistic Regression sample.

```
'accuracy': 0.67,
'recall': 0.7571428571428571,
'precision': 0.7681159420289855,
'f1score': 0.762589928057554
```

However, the model was not tuned and is far too deep. The number of nodes in the model when un-regulated was 18, which is too complicated to be easy to use.

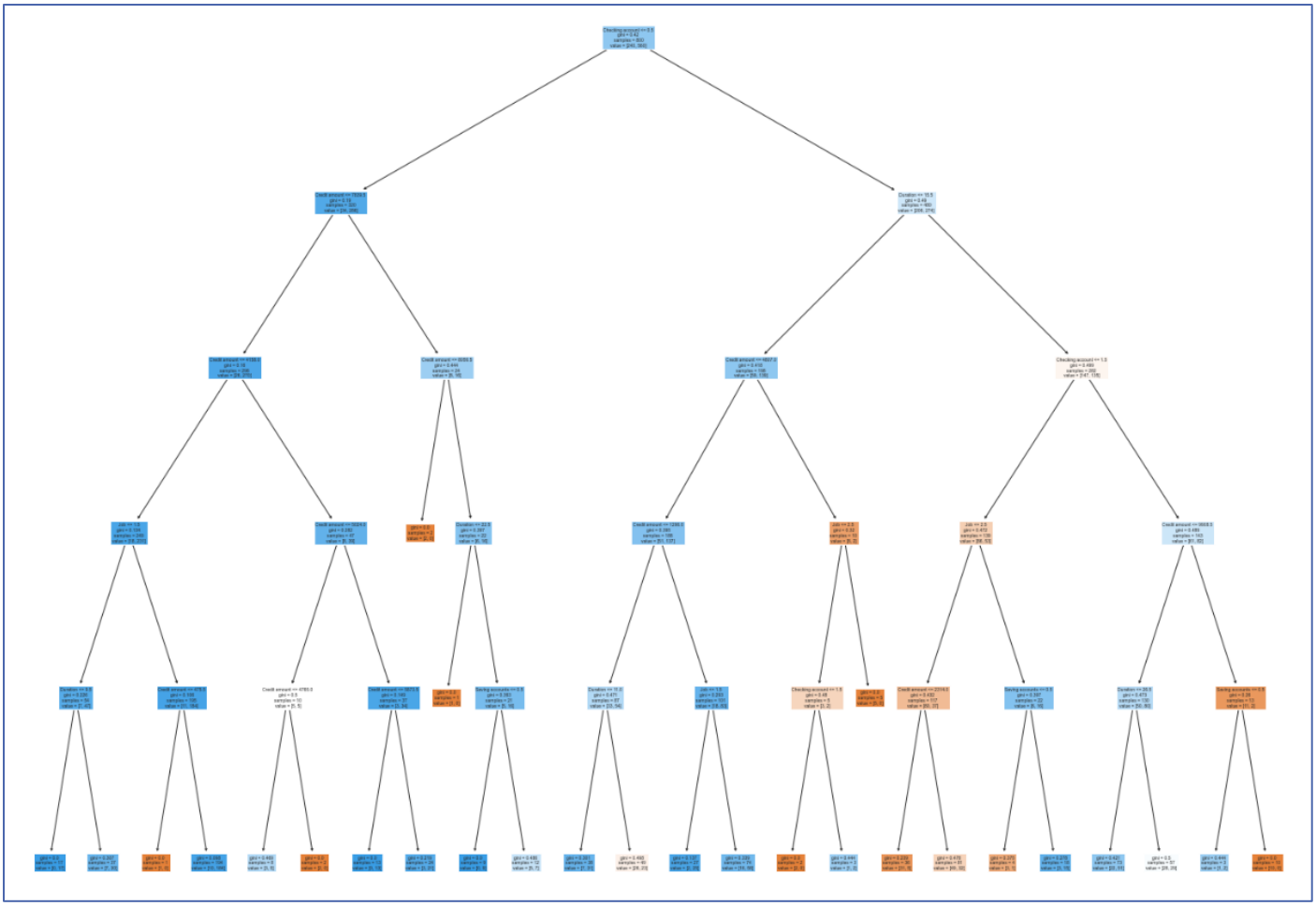


To tune the model GridSearchCV was used to assess the optimal hyperparameters.

```
best_params
    {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1}
```

```
# Trained a customized decision tree
custom_model = DecisionTreeClassifier(criterion='gini', max_depth=5,
min_samples_leaf=1, random_state=rs)
custom_model.fit(X_train, y_train.values.ravel())
preds = custom_model.predict(X_test)
evaluate_metrics(y_test, preds)
# Plot the decision tree
plot_decision_tree(custom_model, feature_names)
```

Newly plotted tree graph shows a much simpler structure which could be used by the business to assess credit risk.



5.0 MODEL PREFERENCE

After reviewing three model types applied to the credit data set, the clear choice for a business to assess credit risk is K-Nearest Neighbour (KNN).

KNN has the highest Accuracy score of all three models (73%) when ***k*** = 7.
It is also a relatively easy model to explain. A new customer can have credit risk predicted based on the values associated with 5 key metrics. Then the overall risk is based on how close the result is to other similar results seen in the training set.

This would be an advantage for a business providing credit to customers, as they would need to be able to clearly articulate why someone may be assessed as a bad credit risk.

6.0 KEY FINDINGS

The analysis took an existing dataset of 1,000 loans provided to customers, along with a series of features pertaining to the customer's financial attributes and the assessed level of risk for each one (Good or Bad).

The purpose of this analysis was to determine the best model for predicting customer credit risk. The intention is for a business to take key features of customer data and determine if they will likely pose a risk if they are provided with a loan.

Data analysis revealed the following:

1. Of the 9 original features in the dataset, only 5 of them contribute to a credit risk result.
2. The main contributing features to a credit risk result are 'JOB', 'CREDIT AMOUNT' and 'DURATION'.

These features and label encoded features SAVINGS ACCOUNT and CHECKING ACCOUNT were the ones chosen for use in the models.

Logistic Regression, K-Nearest Neighbour and Decision Tree Classifier were used to determine which would provide the most accurate result, be explainable to business users and customers, but still be a useful tool for business applications.

Logistics Regression was performed first to establish a baseline level of accuracy for the features used. KNN and Decision Tree were then applied as they are suitable for binary outcomes.

Each model was tested for accuracy, precision, recall and FI-Score to determine which would be the most effective method for assessing which risk category a customer might fall into. They were also run with different hyperparameters to see how that may change the accuracy.

As stated in the previous section, **K-Nearest Neighbour (KNN)** was the most accurate model with the simplest explainability. This was most desirable for a financial institution as it would assist with transparency of the application process.

It should be noted that the peak accuracy was 73%. Which is better than other models, however that still leaves a wide margin for error when assessing credit risk. The KNN model will provide basis for assessing suitability for loans, but it cannot be used as the sole risk classification method.

7.0 NEXT STEPS

Future modelling using this dataset would benefit from further refinements to enhance accuracy. Here are a few possibilities:

- **Include HOUSING feature** - this was initially excluded as it was typically negatively correlated with the results. However, there was one instance where a positive correlation was found with SAVINGS ACCOUNT. Adding this feature into the model could make some improvements to accuracy overall.
- **Increase K** - add additional values of K to see what happens to the accuracy.