Phase 1: Problem Assessment

Background

Globally, the COVID-19 pandemic has had a significant behavioural and psychological impact on individuals. As a result, substantial changes in consumer purchasing patterns have compelled companies to modify their business strategies (Tao et al., 2022). The number of individuals who want to splurge and participate in a post-pandemic buying spree has increased by approximately 51% (Emerging Consumer Trends in a post-COVID-19 World, n.d.), which has led to a significant increase in transactions as observed in Figure 1.

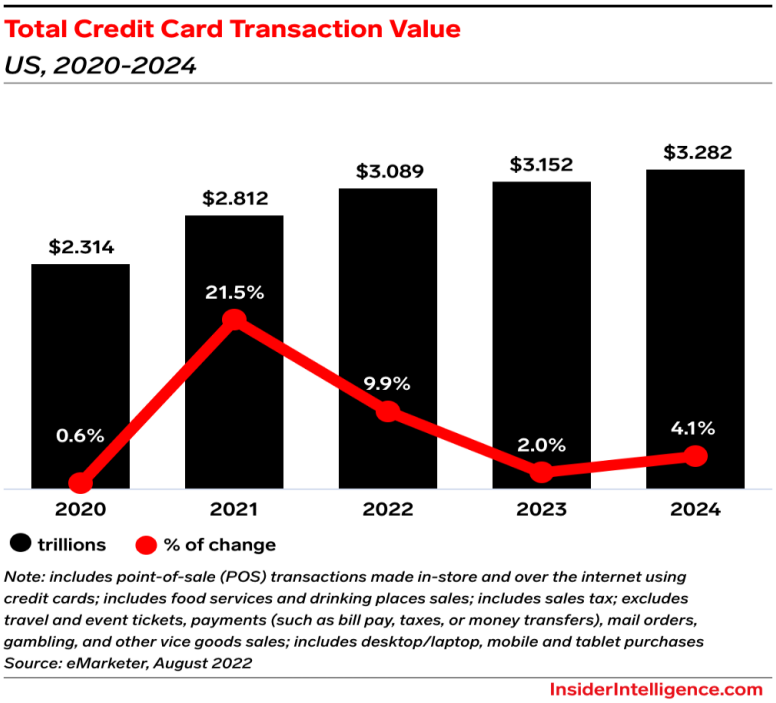

Figure 1: Credit Card Transactions Value (Nunez, 2023)

In December, banks issued 1.37 million new credit cards, an increase of 12% from the previous month. The entire credit card base increased to 68.9 million, marking the highest level in 17 months and a 14.2% increase from the prior year (Ghosh, 2022). Although credit lost popularity to debit as consumers sought to reduce financial risk in the pandemic's early stages, borrowing is again on the upswing. Increases are anticipated to continue, prompting issuers and financial technology (FINTECH) companies to adjust perks and introduce new products (Intelligence, 2023).

Companies have now been presented with the opportunity to influence the next Normal because many of the longer-term shifts in consumer behaviour are still developing. The place and manner in which consumers interact have evolved, and marketing efforts should reflect accordingly (Kohli et al., 2020). Banks and companies must therefore reexamine their business strategies in order to adapt and grow in the post-pandemic era. In addition, banks and businesses should reevaluate their customer base to identify high-value prospects and retain existing customers.

Description of AI Product

The solution designed would aid banks and financial institutions by identifying desired customer traits for various scenarios. Business organisations could also take advantage of this product and integrate it into their process flow by customising and scaling as per the requirements of their respective organisations, industries, and sectors. They could better **identify their top customers' characteristics** and **focus their marketing efforts** on those traits.

The AI product has been created by Machine Learning (ML) **Random Forest Decision Tree (RFDT) algorithm** to build a predictive model. The model would **analyse the desired characteristics of top credit card spenders**. With the traits identified and projected onto ideal prospects, organisations could focus their marketing effort on individuals with specified features. This would reduce costs, increase the acquisition rate for new customers, and retain high-value customers through laser-targeted marketing.

Innovation

Traditionally, this has been mainly adopted by banks and financial institutions. The common application of ML includes algorithmic trading, robo-advising, financial distress forecasting, fraud and compliance, and credit scoring (Buchanan & Wright, 2021). **The solution model assumes a business-centric approach by examining customers' demographic, credit card spendings, repayments and identifies key traits and patterns**.

Top spenders are generally determined by business organisations through domain expert knowledge of the salesman or the analysis of the Pareto principle 80-20 rule. In recent times, visualisation dashboards have enabled organisations to harness the power of data. Dashboards help firms to efficiently assess, monitor and manage their business performance (Kruglov et al., 2021). Businesses would be able to identify unique insights through historical data graphically.

Organisations must utilise these competencies to gain and maintain a sustained competitive advantage (Sen et al., 2021).This model **builds upon historical data and transforms it to predict potential prospects through identified traits**. Businesses could predict the desired customers instead of relying on domain experts and historical infographics, which could be prone to unintentional manipulation by using unsuitable charts or misconceptions.

Phase 2: Data and Knowledge Acquisition

Several machine learning algorithms are frequently employed in the financial industry, depending on the particular goal or problem. These algorithms include:

- **Linear Regression**: Explore relationships between various variables and forecast trends
  Application :Predicting values (eg. stock prices, exchange rates)

  However, linear regression assumes linearity between variables which may not be applicable in most applications. Thus, linear regression will not be considered in this context.

- **Logistic Regression**: Prediction for binary results
  Application: Risk assessments and predictive

  Logistic regression will be discussed in-depth as it is a commonly applied algorithm by many organisations due to its interpretability and ease of use.

- **Random Forest:** Predicting and identifying trends in huge datasets using ensemble learning of random decision trees
  Application: Fraud detection, credit scoring, and risk assessment

  Random forest will be discussed as it is the algorithm of **choice in our AI Product** due to its high accuracy and ability to work with huge data sets due to its robustness and flexibility.

- **Support vector machines (SVM)**: Classify variables on a hyperplane
  Application: Predict share price, fraud detection, and credit scoring

  Support vector is commonly applied in the finance industry. However, the accuracy is lower when compared to RFDT and, therefore, would not be considered.

- **Neural Network**: portfolio optimization, fraud detection, and risk assessment.
  analyze large amounts of data and identify patterns that might be missed

  Although Neural Network may be the most accurate algorithm model of the five, it is not applicable for determining the key variable in a predictive model. Moreover, neural network requires a large dataset and is computationally more expensive which might be applicable in this context. Therefore, neural network will not be discussed in this report.

Logistic Regression (LR)

Logistic regression is a form of statistical model, commonly referred to as the logit model, frequently used in categorization and predictive analytics. Logistic regression analyses the relationship among variable(s) and calculates the likelihood that an event will occur (*What Is Logistic Regression? | IBM*, n.d.).

Logistic regression could take various different forms which include: 1. Binary Logistic Regression (eg: Pass or Fail); 2. Multinomial Logistic Regression (eg: Small, Medium, Large); 3. Ordinal Logistic Regression (eg: Grading from 1 to 5) (Swaminathan, 2019).

With the exception of how they are applied, logistic regression and linear regression are very similar. Whereas logistic regression is used to solve classification difficulties, linear regression is used to solve regression problems (*Logistic Regression in Machine Learning - Javatpoint*, n.d.).

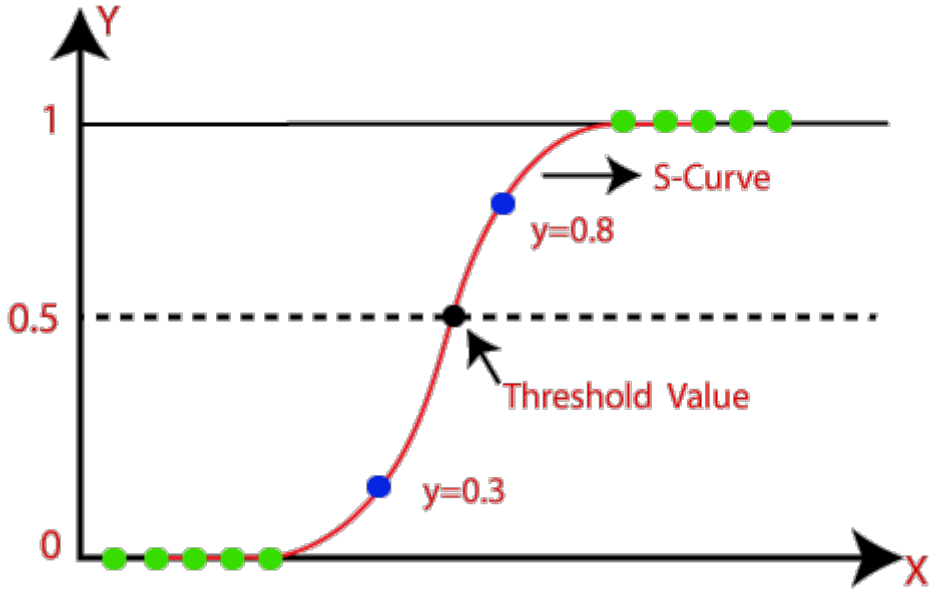

Figure 2: S-Curve (Logistic Regression in Machine Learning - Javatpoint, n.d.)

In logistic regression, an "S" shaped logistic function, which predicts two maximum values, as opposed to a regression line as depicted in Figure 2.

Strength:

- Simple
- Easily interpretable with coefficients
- Could work with non-linear relationships
- Do not require high computational power
- Robust to noise

Limitation:

- Assumes linear relationship
- Sensitive

- Overfitting
- Not suitable for complex relationships:

In summary, logistic regression is a straightforward and efficient approach that may be applied to classification and predictive modelling. However, it might not be appropriate for complex relationships, and additional steps are required to describe non-linear relationships.

Random Forest Decision Tree (RFDT)

Random Forest is an algorithm that utilises the decision tree as its foundation. Like its name suggests, a random forest is made up of several independent decision trees that work together as an ensemble. (Yiu, 2021). This process is repeated several times, with the final forecast being the average of all forecasts made by all decision trees, as depicted in Figure 3.

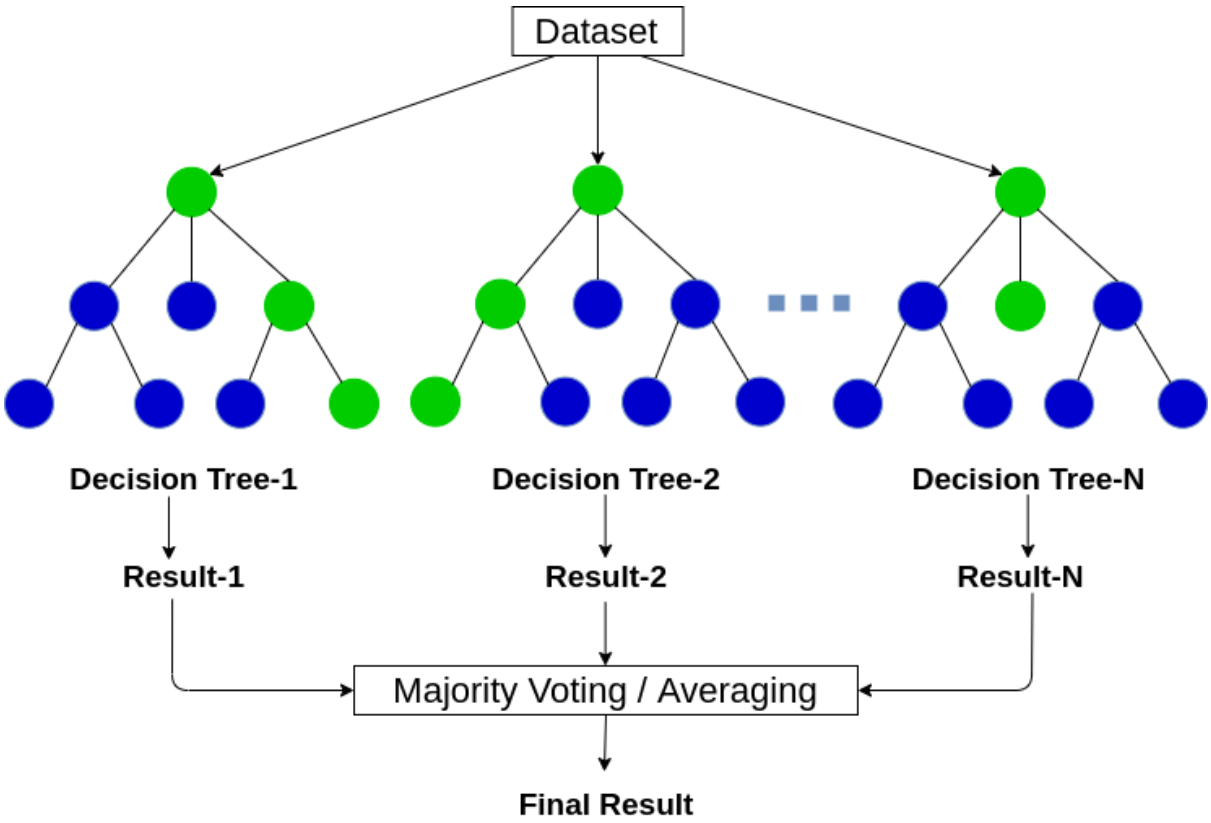

Figure 3: Random Forest Algorithm (Sharma, 2023)

Several techniques for achieving ensemble learning include bagging, boosting, and stacking. According to Wei & Chen (2002), ensemble learning develops a collection of autonomous learners combined by utilising various techniques to develop a strong learner. Due to their tendency to have fewer biases and their ability to reduce overfitting by combining many different individual models, ensemble models produce more accurate predictions.

Strength:

- Increase accuracy and robustness
- Works with high-dimensional datasets
- Prioritize key features automatically
- Resistant to overfitting
- Easy of use

Limitation:

- Not easily interpretable due to complexity
- Might be computationally expensive

Random forest is a powerful tool for predictive modelling, when working with large, complicated datasets. It is commonly applied to a wide range of industries including banking, healthcare, and marketing. Across a wide range of problem areas and practical applications, ensemble systems have shown to be exceedingly efficient and incredibly adaptable (Polikar, 2012).

Therefore, we have elected for Random Forest algorithm due to its diverse applicability and effectiveness. This is critical for organisations where resources might be limited, and decisions could be costly. Moreover, it is able to reduce the prep work required due to its capability to work with robust datasets and a high tolerance for dirty data.

Comparison

Both the prominent machine learning algorithms random forest and logistic regression are employed in predictive analytics. The main variations between the two are as follows:

	Logistic Regression (LR)	**Random Forest Decision Tree (RFDT)**
Approach	Linear	Ensemble
Complexity	Simple	Complex
Interpretability	Simple	Difficult
Linearity	Assumes Linearity	Works with Linear/Non-Linear
Performance	Simple Dataset	Simple & Complex Dataset
Sensitivity	Affected by outliers	Works with missing data

Table A: Algorithm Comparison

The project objectives, dataset size and complexity, and required level of interpretability will often determine the best algorithm for the context.

In this report, the Random Forest would perform better due to the robustness and accuracy of the algorithm. This is because actual datasets in organisations would often not be prepped to perfection, adversely affecting logistic regression models. The skewed results may result in organisations making an alternate decision which could ultimately cost the organisation. Therefore, we have opted to adopt the RFDT as our algorithm of choice.

About the Data

Link: https://www.kaggle.com/datasets/darpan25bajaj/credit-card-exploratory-data-analysis

The dataset was obtained from Kaggle, Credit Card Exploratory Data Analysis, comprising of the following CSV files:

1. **Customer Acquisition**: This table contains details regarding customer information gathered throughout the card issuance cycle. Although the data collected were limited, they make for quite good predictors, such as age, city, income segments, and others.

2. **Spend**: This table presents transaction data from credit card purchases made by customers. It consists of several columns. One of them is the spending type which makes a promising predictor for the classification.

3. **Repayment**: This table provides the credit card repayment done by the customer. It has information on when the customer made the repayment and the repayment amount.

This data set is highly applicable to our use case due to the nature of credit card transactions, and the rise in the adoption of FINTECH payments method post-COVID-19. The data set would allow us to profile the customers from a finance perspective while applying it to business organisations at large. This would synergise with our AI product solution as it would be applicable to both financial institutes for risk management and businesses for an enhanced variation of customer analysis through predictive analytics.

Data Cleaning

The first step after obtaining the dataset was to verify the data's hygiene. Here are some of the steps performed to ensure the data set hygiene and integrity:

- Reviewing the dataset for any missing data
- Check for duplicates
- Identify abnormalities or outliers
- Data format and type standardization

After performing the steps above, this dataset is validated once again to ensure it is ready for use.

Data Pre-Processing

Following this, the data would undergo the Extract-Transform-Load (ETL) stages. One of the preparation steps was to encode the data numerically (Refer to Appendix B). Data encoding is crucial for the ML model to identify patterns and trends in the data. It also decreases dimensionality and assures uniformity of data which reduces the risk of overfitting.

As the dataset comprises of two transactional tables (Spending and Repayment) and a dimensional table (Customer). Pivot aggregation requires tables to be joined. Upon successfully joining the tables, additional features were created for further analysis work:

- Average spending amount
- Average repayment amount
- Average repayment rate
- Card spend ratio

Note: For further information on how this is calculated, refer to Appendix A - Table 1.

Usually, the above steps should be enough for the data pre-processing steps. However, we are still lacking three things, a dependent variable (prediction), a training dataset, and a test dataset. In order to tackle the first issue of the missing dependent variable, we identified the characteristics of a high spender. The approach is to analyze the customer card spending ratio, which takes into consideration the customer's limit and spending amount, with the assumption that the customer card spending ratio is larger than 0.8 (Spends more than 80% of the assigned card's limit).

With regards to the other issue of missing training and test data, this is resolved by splitting the dataset into a 20:80 ratio of test-training data respectively. The 70-30 and 80-20 ratios are the most commonly used threshold for training and test split. However, due to our limited sample size of only 100 rows of unique customers, the 80-20 split was opted to provide for a stronger training model, which would be fed into the RFDT that relies on the average of all the outputs.

Post data pre-processing, the independent (features) and dependent variables are as listed:

- Independent Variables
    - Age
    - City
    - Product
    - Company
    - Segment
    - Average Repayment Rate

- Dependent variable:
    - High Spender: 1=Yes or 0=No

Past Success of Random Forest Applications

Random forest has seen successful application in real-world scenarios. An example is the applications of Random Forest in various industries. Banks and other financial institutes apply RF for fraud detection, and the medical industry uses RF to diagnose the probability of diseases from patients' health data.

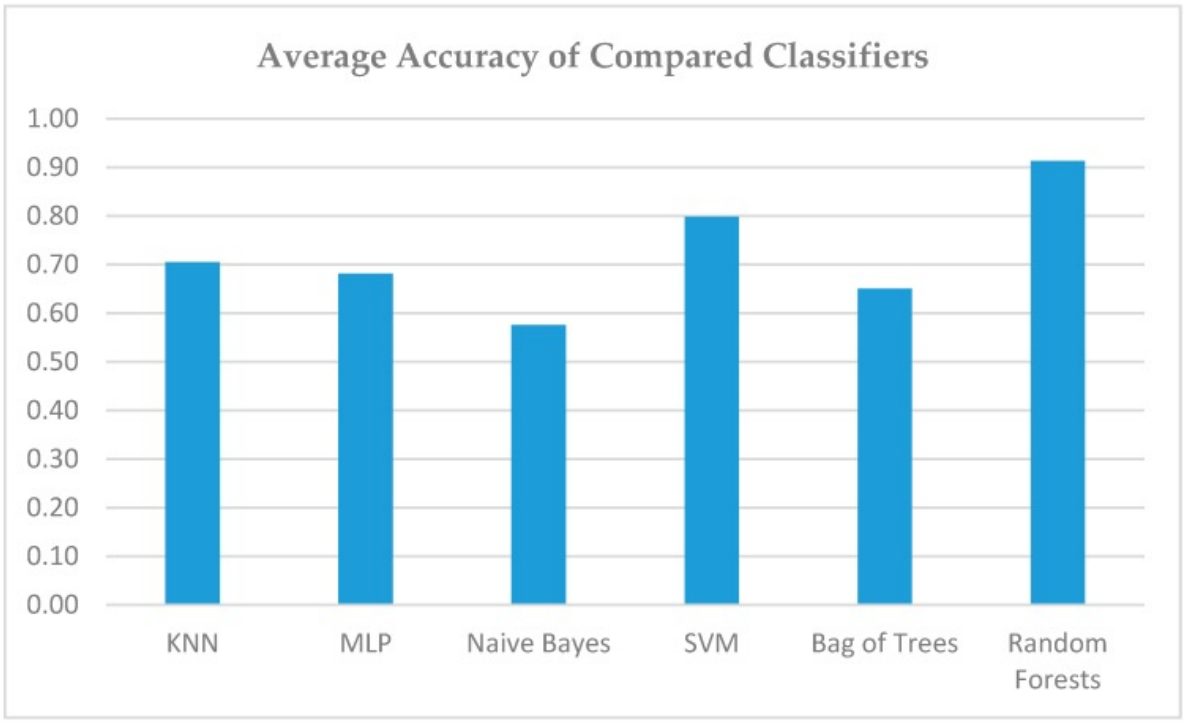

Figure 4: Result of Average Accuracy (Sideris et al., 2019)

A notable success story is the use of random forests on real-world city data for urban planning. According to a study by Sideris et al. (2019), random forest model has the highest accuracy and the highest success rates for key metrics, as depicted in Figure 4 when compared to various other algorithms.

Output

The output of our AI product can take many forms, depending on the specific task and the type of algorithms being used. In this context, there are three outputs produced by the program.

1. The first output is a CSV file with the prediction result after the algorithm determines whether the consumer is a high-spender. As previously stated, this outcome may be utilized for business decisions, risk assessment, etc.

2. There is also a comparison of the Accuracy and unweighted average recall (UAR) of the Decision Tree, Random Forest, and Logistic Regression algorithms. In this context, the decision tree serves as a benchmark. It is critical to examine both accuracy and UAR when evaluating the success of machine learning algorithms since they provide an avenue for the model's performance validation. One measurement may be more relevant than the other depending on the objective and dataset, and the outcome will assist the analyst in determining which algorithm is better suited for the context at hand.

3. Lastly, we can generate the tree plot for Decision Tree and Random Forest, which presents the decision tree visually for enhanced understanding, enabling readers to comprehend the decision-making process efficiently, and the rationale. Each node in the tree represents a specific feature, and the branches define the subsequent route if available.

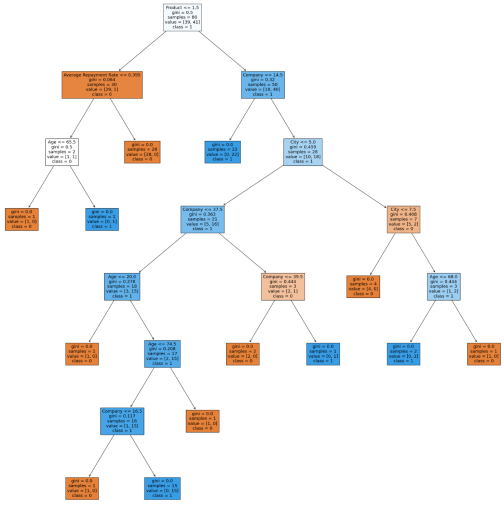

Figure 5: Tree Plot Result from the Decision Tree Model

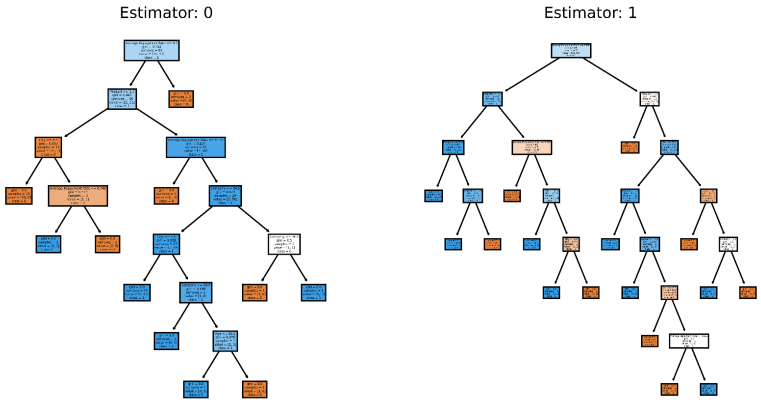

Figure 6: Two Tree Plot Results from the Random Forest Model

```
branata.kurniawan@ITSG008777-MAC ICT619 % python3 prototype.py
----------------------
DT Accuracy: 0.75
DT AUC: 0.76
----------------------
RF Accuracy: 0.85
RF AUC: 0.85
----------------------
LR Accuracy: 0.85
LR AUC: 0.85
```

Figure 7: Comparison Result for DT, RF, and LR on Prediction Accuracy and AUC

Phase 3: Development of Prototype

Here is our journey for creating the prototype in Python:

- We **read the data** with the help of the panda library. Panda reads the CSV files of Customer, Spending, and Repayment and then turns them into DataFrame.
- After all the data is inside the DataFrame, the following action is to **prepare and preprocess the data**. In this step there are several actions performed, which are:
    - Merging of the Customer and Spending DataFrame based on the '*Customer*' variable, then pivoting it to get the average spending amount for each customer with the help of the panda library.
    - Similarly, we performed the same action of merging and pivoting for Customer and Repayment DataFrame to get the average repayment amount.
    - Afterward, we created a new DataFrame based on Customer DataFrame, then moved and mapped the *'Average Spending Amount*' and *'Average Repayment Amount*' for each customer inside the DataFrame.
    - And then, using the newly merged DataFrame, we created the rest of the variables with the help of a numpy library. The Numpy library provides functionality to make working with multi-dimensional arrays easier. Here are the variables, '*Card Spend Ration*', '*High Spender*', and '*Average Repayment Rate*'.
    - Lastly, panda provided this mapping functionality that extended to the DataFrame, which helped easily encode the data into the format we provided (Refer to Appendix B for the data encoding information).
- Next is **splitting the data** into the train and test data set with the assumed test size and random state. We achieve the split by using the function called '*train_test_split*' from sklearn.model_selection library.
- We also **created two variables**, 'independentVarColumnName' (contains the independent variables' column name) and 'dependentVarColumnName' (contains the dependent variable's column name), to reuse throughout the code.
- Using the two new variables above, we used the '*loc*' function from panda to **extract the train and test data** based on the column names of the independent and dependent variables specified. This extraction will separate the independent and dependent variables for trains and test DataFrame.
- Furthermore, we will **choose an AI model** based on the machine learning algorithms after all the above actions. For,
    - Decision Tree (DF) → '*DecisionTreeClassifier*' model from sklearn.tree library.
    - Random Forest (RF) → '*RandomForestClassifier*' model from sklearn.ensemble library.
    - Logistic Regression (LR) → '*LogisticRegression*' model from sklearn.linear_model library.
- Next is to **train the model** by fitting it with the trained DataFrame using independent

and dependent data.

- After finishing the training, we proceed to **test the model**. We need to call the '*predict*' function to test the model and pass the independent test variable DataFrame into it. The function then returned with the prediction results.
- For this prototype, **prediction results are added into the DataFrame** as the new column named '*Prediction Result*'. Subsequently, DataFrame will be saved into the local folder as one of the outputs.
- Also, by using the prediction result object, we **calculate the prediction accuracy and Unweighted Average Recall (UAR)** of each algorithm by using the '*accuracy_score*' (for accuracy) and '*roc_auc_score*' (for UAR) function from sklearn.metrics library. The results will then be consolidated into a single table and saved as an image as one of the outputs.
- Lastly, we **generated the decision tree plot** for DT and RF based on the trained model as one of the outputs.

Phase 4: Development of a Complete System

The initial prototype development phase of the project covered most of the fundamental development work for the AI model to achieve basic functionality. Therefore, in this project component, the main emphasis and effort will be on model enhancement and evaluation. This is to improve the effectiveness and efficiency of the model algorithm and output. Model accuracy is essential for financial institutions and businesses to integrate it into their systems for critical decision-making. The model improvement will be divided into the following stages:

1. Additional Performance Metrics

New additional performance metrics from Python Library (*sklearn.metrics*) are added to enhance model performance:

1. Mean Squared Error (MSE): calculated by using the function *`mean_squared_error`*
    - Calculates the average squared difference between the target variable's expected and real values
    - Greater accuracy in predicting the objective variable is indicated by a lower MSE

2. Precision rate: calculated by using the function *`precision_score`*
    - Precision = True Positives / (True Positives + False Positives)
    - Of all the positive forecasts, it calculates the percentage of accurate positive predictions
    - An increased precision rate typically means that the algorithm is producing fewer false optimistic forecasts

3. Recall (Sensitivity) rate: calculated by using the function *`recall_score`*
    - Recall = True Positives / (True Positives + False Negatives)
    - Percentage of genuine positive cases the algorithm accurately finds among all of the dataset's positive instances
    - High recall rate indicates that most of the positive cases in the dataset can be accurately identified. False positives, however, could arise from erroneous categorization.

Although each performance are strong indicators in their own rights, it is crucial to remember that each performance measure should always be taken into account in conjunction with other performance metrics. This will help us comprehend the model's results more thoroughly.

2. Estimation performance boost with N-Fold Cross Validation

An approach frequently used in machine learning to assess a model's success is N-fold cross-validation. The accessible dataset is divided into N equal-sized "folds" before N rounds of training and testing.

N-Fold Cross Validation

```
# Preparing the k-fold cross validation with 5 splits
nSplits = 5
kf = KFold(n_splits=nSplits, shuffle=True)

# Create the array to contains the scores
dtAccuracyTestScores = np.zeros(nSplits)

for i, (trainIdx, testIdx) in enumerate(kf.split(newMergedDf)):
    trainDf = newMergedDf.loc[trainIdx]
    dtTestDf = newMergedDf.loc[testIdx]

    # Train the model
    ...

    # Predict using the trained model
    ...

    # Calculate the test scores and saved it to and array
    dtAccuracyTestScores[i] = accuracy_score(dependentTestDf, dtPredictionResult)

# Find the mean of the score
dtData[dtTitle] = [
   np.mean(dtAccuracyTestScores)
]
```

We implemented the N-Fold Cross Validation with a value of 5 with the help of the `KFold` function from the `*sklearn.model_selection*` library. 5, 10, or 20 are typical values for N. 5 will be selected for N due to the consideration of our limited data samples.

We divided the sample data into five equal subsets for training and testing, with each set tested once. After which, the results of each iteration are collated to estimate the model's overall performance.

Therefore, N-fold cross-validation can provide a more reliable estimate of the model performance as it utilises all available data and reduce the possibility of overfitting.

3. Decision Tree Pruning

As the prototype for both the Random Forest Decision Tree and generic Decision Tree performed above expectations during the development of the prototype. We set out to further improve the model by exploring various metrics and reducing the complexity by limiting the depth of the decision tree models. This would aid in avoiding overfitting, which can lead to detrimental performance.

Prunning - With the example of Random Forest

```
# Creating a random forest model with max_depth 10 to reduce its complexity
model = RandomForestClassifier(max_depth=10)
```

Pruning simplifies the structure by removing necessary nodes or branches where we define the 'max_depth' during the model creation process. Essentially, we remove the unnecessary branch and reduce the size of the decision tree, which also hasten the training time and reduce computational requirements.

4. Output Improvement for Result Comparison

Table Comparison Generation

```
# Create an array to contain result
rfData = {
    'RF': ['Accuracy', 'AUC', 'MSE', 'Precision Rate', 'Recall Rate'],
}

# Defined an array for max depth to test
maxDepth = [-1,20,15,10,7,6,5]
for num in maxDepth:
    # Create the model with the depth
    ...

    # Train the model
    ...

    # Test the model and get the score
    ...

    # Consolidate the score and insert it to the array above
    rfTitle = "Default" if num == -1 else ("MaxDepth=" + str(num))
    rfData[rfTitle] = [
        np.mean(rfAccuracyTestScores),
        np.mean(rfAucTestScores),
```

```
        np.mean(rfMseTestScores),
        np.mean(rfPrecisionTestScores),
        np.mean(rfRecallTestScores)
    ]

# Generate the table into an image
rfDf = pd.DataFrame(rfData)
dfi.export(rfDf,"Random Forest Output/RF pruning result comparison.png")
```

To facilitate the consumption of results for everyday users, the results are compiled into a PNG format for easy understanding through graphical depiction. This is a step up from results generated in the terminal which could be complicated and inaccessible to business users or those outside the specialised domain. This report reflects all the metrics by generating a table that compares Accuracy, AUC, MSE, precision rate, and recall rate for all the maximum depths specified in the code. The information could be easily understood by representing them in a table format that is highly similar to a spreadsheet used by business professionals.

5. Model Generation for Business Process Integration

Model Generation or Load the Model - Using Pickle

```
# Generate the model
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)

# Load the model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)
```

Similarly, the model is processed into a '.pkl' package by applying the' pickle' library. This package is a binary file containing the model, which will be integrated into the daily operation process to perform in the intended use case of the business organisation. The compilation reduces the need for repetitive coding, which may not be performable by those outside the specialised domain. With the compiled package, operational users could execute the package generated into the business process and perform prediction analysis without needing expert domain knowledge.

Phase 5: Evaluation and revision of the System

System Revision

After performing enhancement modifications to the AI model, the next step would be to evaluate if the modifications has enhanced the algorithm of our AI model. Utilising the various performance metric, the result will serve as a benchmark and control to evaluate the model.

As mentioned in Phase 4, addition metrics were included to the original metrics. The new performance metric benchmark would comprise of the following:

1. Accuracy: At least 80% or higher of accuracy score
   This would ensure that the model is able to functional as intended, and correctly identify the target (characteristics of high spenders).

2. Area Under Curve (AUC)
   Fair: 0.7 to 0.8 is
   Good: 0.8 to 0.9
   Excellent: > 0.9

   In our model, the aim would be to achieve "fair" or "good" to ensure overall performance.

3. MSE: A lower MSE score indicates better performance
   The aim would be to achieve a maximum MSE of 0.2.This is to ensure that the model would not deviate significantly and result in erroneous results with misdirected efforts or costly implication.

4. Precision Rate & Recall Rate: Highest possible rate
   Precision and recall rate is essential as our AI model adopts predictive analytics. A higher precision rate (i.e., a low false positive rate) precedes the recall rate. This is due to opportunity cost, as falsely identifying an individual as a high-spender could still generate value for the organisation compared to missing out on a high-spending prospect (false negative), which would often be more costly.

Model Evaluation

	LR	Default
0	Accuracy	0.830000
1	AUC	0.521429
2	MSE	0.470000
3	Precision Rate	0.316923
4	Recall Rate	0.500000

Figure 8: Logistic Regression Result Output

Examining the result of the Logistic Regression Model in Figure 8 above to the respective performance metrics benchmarks. It was observed that the LR's accuracy is relatively high at 0.83. However, the MSE depicts a different picture with a score of 0.47. LR's precision and recall rate is also lower, with scores of 0.32 and 0.5, respectively. Therefore, using these results, we conclude that the LR model will not fit our AI Product well.

	DT	Default	MaxDepth=6	MaxDepth=5	MaxDepth=4	MaxDepth=3	MaxDepth=2	MaxDepth=1
0	Accuracy	0.730000	0.830000	0.780000	0.850000	0.850000	0.860000	0.880000
1	AUC	0.729638	0.828189	0.786630	0.836854	0.844032	0.853590	0.876970
2	MSE	0.270000	0.170000	0.220000	0.150000	0.150000	0.140000	0.120000
3	Precision Rate	0.718803	0.842735	0.757929	0.881111	0.909206	0.944444	0.975000
4	Recall Rate	0.796883	0.781313	0.763370	0.755931	0.743834	0.747179	0.772121

Figure 9: General Decision Tree Result Output

Next, we will examine the General Decision Tree in Figure 9. The result table generated provides the basis for comparison of the algorithm performance. As mentioned in the previous section, maximal depth has been considered for decision tree models to reduce the complexity that could lead to potential complications (e.g. overfitting). The various variables determine the maximum depth of the Decision Tree model. The simulation utilises the default *max_depth* (by not adding max_depth parameters when creating the model) and the *max_depth* 1 through 6. Based on the result above, *max_depth* 1, 2, 3, and 4 generate good accuracy with a high precision rate, high recall rate, and low MSE. However, those are not optimal due to underfitting, where the Decision Tree model is oversimplified. Therefore, from the results above, the max_depth of 6 yields the best performance with high accuracy, precision rate, and recall rate while having a low MSE score.

	RF	Default	MaxDepth=20	MaxDepth=15	MaxDepth=10	MaxDepth=7	MaxDepth=6	MaxDepth=5
0	Accuracy	0.860000	0.850000	0.860000	0.880000	0.860000	0.850000	0.850000
1	AUC	0.856633	0.865865	0.863214	0.874621	0.860152	0.838195	0.852399
2	MSE	0.140000	0.150000	0.140000	0.120000	0.140000	0.150000	0.150000
3	Precision Rate	0.928247	0.911111	0.898413	0.955556	0.915556	0.917949	0.915000
4	Recall Rate	0.773651	0.795833	0.800000	0.785606	0.773636	0.759507	0.772980

Figure 10: Random Forest Decision Tree Result Output

Like the general Decision Tree, pruning was performed on the Random Forest model by limiting the max depth to reduce complexity. The Random Forest model differs from the general Decision Tree model in determining the number of maximum depth. This is due to other hyperparameters affecting the number of branches in a Random Forest model, which includes sample size (min_samples_split) and the maximum number of features (max_features). In our test, various max_depth of 20, 15, 10, 7, 6, and 5 are tested due to the limited sample size. From Figure 10 above, most results provide similar results except for max_depth 10, which comprises the highest accuracy, precision rate and lowest MSE score. Therefore, the Random Forest Decision Tree with a max_depth of 10 yields the best result compared to the other models.

Phase 6: Integration and Maintenance of the System

Model Integration

The integration of the system could be easily adopted into any business organisation. By integrating this script package into any workflow process or periodic report generation, business/organisation could easily utilise this model

The integration of the AI Model should be easily integratable into any organisation's operation process. Below are a few integration methodology to be considered:

1. **APIs:** The API enables the model to communicate with other software elements or services and receive data for processing.
   Python models can profit from using APIs for integration in terms of freedom, scaling, version control, and security, but it also adds complexity, overhead, costs, and constraints. Therefore, defining a precise extent is crucial to avoiding extra costs.

2. **Libraries**: Make it possible to incorporate the model into already-existing apps. For instance, if the company employs Java-based software, it could incorporate Python models using the Jython library.
   Libraries make things simple. However, this comes with constrained capabilities. If the intention was to grow up in the near future, organizations need to take this into account.

3. **Cloud services**: Companies can install and incorporate machine learning models in the cloud using services provided by cloud platforms like Amazon Web Services (AWS) and Microsoft Azure.
   Cloud services might be the more affordable option when compared to other connection methodologies. But the primary issue would be the safety and anonymity of the submitted data. Furthermore, the company would be reliant on the host's server as latency and the vendor's service availability could affect the business if system failure were to occur.

4. **Workflow automation**: To incorporate Python models into their operations, some companies may use workflow automation tools.
   Workflow automation can provide advantages like effectiveness, reliability, scale, and personalization. When changes are necessary, different aspects of the business method may be impacted. This is by nature expensive and complicated. In addition, opportunity costs might start to mount during the repair time.

As highlight in phase 4, among the outputs generate, the model is created as a binary file. This model generated is crucial for the integration process. The recommended integration methodology for the system are as follows:

- The primary recommendation is to integrate the model through an API (Application Programming Interface). The drawback is the start up cost for development and deployment of the API. After development, users can use the model by calling the API through some HTTP request passing in the necessary data as the request's body. But this integration plan provides a very significant benefit that is expandable, which means that multiple applications can use this model for the prediction and are not limited to mobile or web applications.

- The alternate recommendation would be deploying the model to a repository where new data are stored for workflow automation. The data will be pushed into the AI product to generate predictions based on defined triggered settings (e.g. Daily 12 AM). This may involve consuming the model on mobile or web applications, where the model will be required to be part of the application package. The integration process could be efficiently executed and begin operations. However, it should be noted that the application codebase size could increase if not monitored or has limitations set in place.

In summary, we would recommend API or workflow automation depending on the organisation's requirement and strategic plan.

Model Maintenance

After development and deployment, maintaining the model is crucial in ensuring that the model improves over time and performance reliably. Therefore, to ensure that the optimal model performance, the maintenance plans are as follows:

- Set up a system to monitor the model's performance and alert triggers to identify when the model's performance falls below a specific level, detect anomalies and system failure

- Retrain the model regularly with new data to ensure that it continues improvement to the everchanging scenario. The option would be to :
  1. Set up a pipeline to automatically retrain the model
  2. Manually retrain the model as needed

- Track the model's performance over time, and utilize the data to analyze and improve the model continually. This may entail experimenting with various algorithms, hyperparameters, or features and employing approaches such as A/B testing to assess the efficiency of our implemented enhancements.

Appendices

Appendix A. Data Dictionary & Terminology

There are three CSV files used as part of the dataset, and here are the tables explaining each of the files and providing an explanation for its variables:

Table 1

Data dictionary for Customer Acquisition CSV file

Column Name	Type	Description
No	Integer	Unique identifier for customer acquisition table
Customer	String	Customer ID which is the identifier for each unique customer
Age	Integer	Customer's age
City	String	City where customer reside
Product	String	Class category of card owned by the customer
Limit	Float	Customer's spending limit
Company	String	Company where customer work
Segment	String	Customer's Employment type
[Generated] Average Spending Amount	Float	Customer’s average spending amount. Generated the data by merging and pivoting the customer & spend data. And then averaged the spending amount per customer.
[Generated] Average Repayment Amount	Float	Customer’s average repayment amount. Generated the data by merging and pivoting the customer & repayment data. And then averaged the repayment amount per customer.
[Generated] Average Repayment Rate	Float	Customer’s average repayment rate. Calculated

		by dividing *Average Repayment Amount* with *Average Spending Amount*
[Generated] Card Spend Ratio	Float	Customer's card spend ratio. Calculated by dividing *Average Spending Amount* with *Limit*
[Generated] High Spender	Integer	A marker to mark whether the customer is a high-spender ('1') or not a high-spender ('0'). Generated by comparing if *card spend ratio* is more than 0.8 (based on assumption)

Table 2
Data dictionary for Spend CSV file

Column Name	Type	Description
SI No	Integer	Unique identifier for spend table
Customer	String	Customer ID which is the identifier for each unique customer
Month	Date (DD-MMM-YY)	Date when the spending occurred
Type	String	Category type where the spending belong to
Amount	Float	Amount of the spending

Table 3
Data dictionary for Repayment CSV file

Column Name	Type	Description
SL No	Integer	Unique identifier for repayment table
Customer	String	Customer ID which is the identifier for each unique customer
Month	Date (DD-MMM-YY)	Date when the repayment occurred

Amount	Float	Amount of the repayment

Appendix B. Data Encoding Information

Here are the data encoding information for City, Product, Company, and Segment:

Table 4

Data mapping for column City

Column Name	City
Data Type	String
Encode	BANGALORE → 1 CALCUTTA → 2 COCHIN → 3 BOMBAY → 4 DELHI → 5 PATNA → 6 CHENNAI → 7 TRIVANDRUM → 8

Table 5

Data mapping for column Product

Column Name	Product
Data Type	String
Encode	Gold → 1 Silver → 2 Platinum → 3

Table 6

Data mapping for column Company

Column Name	Product
Data Type	String
Encode	C1 → 1 C2 → 2 C3 → 3 …….. C39 → 39 C40 → 40 C41 → 41

Table 7

Data mapping for column Segment

Column Name	Segment
Data Type	String
Encode	Self Employed → 1 Salaried_MNC → 2 Salaried_Pvt → 3 Govt → 4 Normal Salary → 5

Appendix C. User Manual

Source Code
Our source code consists of 2 python files:

1. **generate_model.py**

This python file's main functionality is to generate a model that will be used to classify the customer into a high spender group or not. The other functionality is to generate images that contain the table with the performance metrics for 3 models to help us evaluate and compare which model to use (which is the Random Forest Model with max depth of 10).

2. **run_model.py**

And next, this python file contains the code which used the model generated by the first python file to process the actual dataset and generate the CSV file with the prediction result as part of it. The code also generates the first 5 trees out of 100 of the Random Forest that can explain the importance of each variable.

Requirement
To run the source code, here is the requirement:

- To have python version 3.11.2 installed. Please follow this link right here to install the python if you do not have it installed: Windows, and MacOS

Installation
To run both of our python files there are several libraries that need to be installed using the package manager `pip`. Here are the commands to install the required libraries:

```
pip install numpy
pip install pandas
pip install sklearn
pip install tqdm
pip install dataframe_image
pip install pickle
pip install matplotlib
```

Run the code
And lastly after we are done with the libraries installation, the last step is to run the code. Here is the command to run the generate_model.py:

```
python3 generate_model.py
```

Please wait a few seconds for the code to finish the process. Once you see the message `Done!` and the progress bar is 100% that means the code successfully finished generating the model and other outputs. The model will be in the parent directory with file name `***model.pkl***`.

And once the model generated then you can run the model by using the following command:

```
python3 run_model.py
```

The following command will create an output directory which will contain the CSV file with the prediction in it and an image that shows the first 5 trees from the Random Forest model that we generated.

References

Buchanan, B. G., & Wright, D. (2021). The impact of machine learning on UK financial services. *Oxford Review of Economic Policy*, *37*(3), 537–563. https://doi.org/10.1093/oxrep/grab016

Emerging consumer trends in a post-COVID-19 world. (n.d.). McKinsey & Company. https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/emerging-consumer-trends-in-a-post-covid-19-world

Ghosh, S. (2022, March 4). How credit card spends connected to consumer confidence | Mint. *Mint*. https://www.livemint.com/industry/banking/credit-card-spends-point-to-rising-consumer-confidence-11646332519120.html

Intelligence, I. (2023, January 24). *Credit Card Industry in 2023: Analysis and Trends in Payment Processing*. Insider Intelligence. https://www.insiderintelligence.com/insights/credit-card-industry/

Kohli, S., Timelin, B., Fabius, V., & Moulvad Veranen, S. (2020). How COVID-19 is changing consumer behavior - now and forever. *McKinsey & Company*. https://www.mckinsey.com/~/media/mckinsey/industries/retail/our%20insights/how%20covid%2019%20is%20changing%20consumer%20behavior%20now%20and%20forever/how-covid-19-is-changing-consumer-behaviornow-and-forever.pdf

Kruglov, A., Strugar, D., & Succi, G. (2021). Tailored performance dashboards—an evaluation of the state of the art. *PeerJ Computer Science*, *7*, e625. https://doi.org/10.7717/peerj-cs.625

Logistic Regression in Machine Learning - Javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/logistic-regression-in-machine-learning

Nunez, A. (2023, January 17). *Banks post consistent card volume growth in Q4, but consumer spending remains uncertain in 2023*. Insider Intelligence. https://www.insiderintelligence.com/content/banks-q4-data-shows-consistent-volume-growth-though-consumer-spending-remains-uncertain-2023

Polikar, R. (2012). Ensemble Learning. *Ensemble Machine Learning*, 1–34. https://doi.org/10.1007/978-1-4419-9326-7_1

Sen, J., Sen, R., & Dutta, A. (2021). Introductory Chapter: Machine Learning in Finance-Emerging Trends and Challenges. *IntechOpen EBooks*. https://doi.org/10.5772/intechopen.101120

Sharma, A. (2023, March 3). *Random Forest vs Decision Tree | Which Is Right for You?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Sideris, N., Bardis, G., Voulodimos, A., Miaoulis, G., & Ghazanfarpour, D. (2019). Using Random Forests on Real-World City Data for Urban Planning in a Visual Semantic Decision Support System. *Sensors*, *19*(10), 2266. https://doi.org/10.3390/s19102266

Swaminathan, S. (2019, January 18). *Logistic Regression — Detailed Overview - Towards Data Science*. Medium. https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

Tao, H., Sun, X., Liu, X., Tian, J., & Zhang, D. (2022). The Impact of Consumer Purchase Behavior Changes on the Business Model Design of Consumer Services Companies Over the Course of COVID-19. *Frontiers in Psychology*, *13*. https://doi.org/10.3389/fpsyg.2022.818845

Wei, X., & Chen, Y. (2022). Early Warning Model for Financial Risks of Listed Companies Based on Machine Learning. *2022 International Conference on Machine Learning*

and Intelligent Systems Engineering (MLISE). https://doi.org/10.1109/mlise57402.2022.00100

What is Logistic regression? | IBM. (n.d.). https://www.ibm.com/sg-en/topics/logistic-regression

Yiu, T. (2021, December 10). *Understanding Random Forest - Towards Data Science*. Medium. https://towardsdatascience.com/understanding-random-forest-58381e0602d2