ALBUKHARY INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING AND INFORMATICS

YEAR 2 SEMESTER 2 2024/2025

CCS2213

Machine Learning

Individual Assignment

TITLE	**Start-up Success Prediction**	
Name	**Muhammad Zulfan Abidin**	**AIU22102088**

Table of Contents

1.0 Dataset Background

Startups play a vital role in today's economy. Compared to large, established companies, smaller and younger businesses create jobs at a much higher rate. They also drive innovation and push industries forward by introducing fresh ideas and increasing competition. (Sharchilev et al., 2018). Investing in early-stage startups comes with a lot of uncertainty. Right now, the tools investors use aren't strong enough to properly assess risks or manage unpredictability, making it harder to identify promising opportunities. (Lotfalia et al., 2022).

The methodology begins with data preprocessing, which includes handling missing values, selecting the most relevant features, and applying normalization to enhance model efficiency. (Nikhitha et al., 2025). The dataset is then divided into training and testing subsets, ensuring a robust evaluation of the model's predictive capabilities. To build the predictive framework, multiple classification algorithms are utilized, including **K-Nearest Neighbors (KNN), Decision Tree, and Random Forest.** Each model is assessed using key performance metrics such as **accuracy score, hold-out validation, classification report**. Additionally, **data visualization tools** like Matplotlib and Seaborn are employed to extract meaningful insights into data distribution and model performance.

a. Background Study of The Dataset

   Startups have become a key driving force behind innovation and economic growth worldwide. However, research suggests that around **90% of startups fail**, making it essential for investors, financial advisors, and governments to identify the **10% that will thrive**. These successful startups have the potential to generate **higher returns, boost revenue, and contribute significantly to the economy**. Based on the literature review titled "Predicting Startup Success" at 1st International Conference on Science and Technology Innovation (ICoSTEC) by **Harjo Baskoro et al.** explores the **critical success factors** that can be used to build a predictive model using machine learning algorithm to predict the success of a startup. (Baskoro et al., 2022)

b. Class Distribution of the Dataset

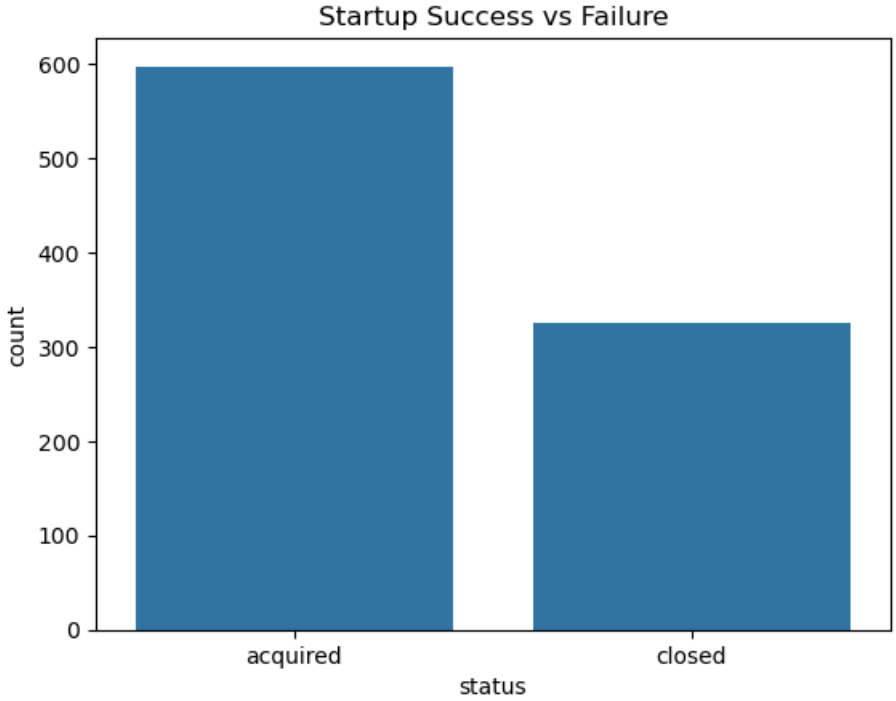
   In the given dataset, the target variable **"status"** represents startup success, with two possible outcomes: **"acquired" (1)** and **"closed" (0)**. The class distribution is significantly imbalanced, with **597 acquired startups (64.7%)** and **326 closed startups (35.3%)**. This imbalance suggests that the dataset favors successful acquisitions, which may lead to biased model predictions, disproportionately optimizing for the dominant class while failing to generalize for underrepresented failed startups. As a result, the model might struggle to identify meaningful patterns that distinguish failing startups.
   To mitigate this issue, **resampling techniques** such as **Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples of closed startups, random under-sampling to reduce the overrepresented acquired startups, or class weighting to penalize misclassification** can be applied. These techniques aim to **enhance**

prediction fairness and ensure balanced learning, making the model more effective across both startup outcomes. To address this issue, **class weighting** has been applied during model training. By adjusting the weights assigned to each class, the model ensures that misclassification of the minority class (**closed startups**) is penalized more heavily, improving prediction fairness and recall. This technique enhances learning without modifying the dataset, making it a practical approach for addressing moderate class imbalance. Proper evaluation metrics such as **precision, recall, and F1-score** will be essential to accurately measure the model's performance on the imbalanced data.

Figure 1a.

2.0 Pre-processing options

Effective data pre-processing is crucial for building robust and accurate machine



learning models. In this project, several pre-processing steps were implemented to ensure the dataset was suitable for classification tasks and to enhance model performance:

1. Handling Missing Values

The dataset contained missing values in several columns. To address this, all missing values were imputed using the **mean** of each column. This approach preserves the dataset size and is appropriate for numerical features, ensuring that no data is lost due to missing entries.

```
df_encoded = df_encoded.fillna(df_encoded.mean())
```

2. Encoding Categorical Variables

Since machine learning models require numerical input, all categorical columns were converted to numerical format using **one-hot encoding**. This

method creates binary columns for each category, allowing the models to interpret categorical information without imposing ordinal relationships.

df_encoded = pd.get_dummies(df, drop_first=True)

3. Target Variable Transformation

The target variable, status, was transformed into a binary column called status_binary, where 1 represents "acquired" (successful startup) and 0 represents other statuses. This transformation is essential for binary classification.

df["status_binary"] = df["status"].apply(lambda x: 1 if x == "acquired" else 0)

4. Feature Selection

To ensure the reliability and generalizability of our machine learning models, we performed careful feature selection. Initially, feature importance was determined using a Random Forest Classifier. However, during this process, we identified two features—**labels** and **status_closed**—that were highly correlated with the target variable and contained information that would not be available in a real-world prediction scenario.

These features were considered **data leakage** because they directly or indirectly revealed the outcome (status_binary). Including them in the model resulted in unrealistically high accuracy, which would not generalize to new, unseen data.

Therefore, both labels and status_closed were dropped from the feature set before model training and evaluation. This step ensures that the model's performance reflects its ability to learn from genuine predictive features, rather than from information that would not be available at prediction time.

By removing these leaking features, we improved the validity of our feature selection process and obtained more trustworthy model evaluation results.

5. Data Splitting and Cross-Validation

The dataset was split into training and testing sets for hold-out validation, and **stratified cross-validation** was used to ensure robust evaluation, especially given the imbalanced nature of the target variable.

6. Class Weighting

To address class imbalance, class weighting was applied in the classifiers. This ensures that the minority class receives appropriate attention during model training, improving the model's ability to predict rare events.

3.0 Model Evaluation Technique

Selecting an appropriate model evaluation technique is crucial for obtaining reliable and unbiased estimates of a machine learning model's performance, especially when dealing with imbalanced datasets such as startup success prediction.

a. Chosen Technique: Hold-Out Validation

In this project, **hold-out validation** was selected as the primary model evaluation technique for both **Decision Tree** and **K-Nearest Neighbors (KNN)** classifiers. This method involves splitting the dataset into separate **training** and **testing** sets, ensuring that the model is evaluated on unseen data. Specifically, a **80-20 split** was used, where 80% of the data is utilized for training, and 20% is reserved for testing.

b. Reasons for Choosing Stratified Cross-Validation:

1. Simplicity & Computational Efficiency: Hold-out validation is straightforward to implement and computationally less intensive than cross-validation, making it ideal for faster experimentation.
2. Avoids Data Leakage: Since the test set remains separate from training, the evaluation reflects real-world performance without overfitting risks.
3. Reliable Performance Estimate for Large Datasets: When the dataset is sufficiently large, a well-proportioned hold-out split provides a reasonable performance estimate without requiring repeated training iterations.Comparison with Hold-Out Validation.

c. Challenges with Imbalanced Data

One key challenge in using hold-out validation on an imbalanced dataset is that the test set may not contain a balanced representation of classes. In this dataset, the target variable "status_binary" is distributed as 597 acquired startups (64.7%) and 326 closed startups (35.3%), meaning the model could favor the majority class (acquired startups) and struggle to correctly classify minority instances (closed startups).

d. Solutions Implemented

1. Class Weighting Applied → To counteract the imbalance, class weighting was used in both models to ensure the minority class (closed startups) is given higher importance, improving recall.
2. Evaluation Using Precision, Recall & F1-Score → Instead of relying solely on accuracy (which can be misleading in imbalanced datasets), performance was measured using precision, recall, and F1-score to assess how well the model handles both classes.

e. Comparison with Stratified Cross-Validation

While stratified cross-validation ensures balanced class distribution across multiple folds, hold-out validation was chosen for simplicity and efficiency. Although a single train-test split may introduce variance in results, applying proper class weighting and targeted evaluation metrics mitigates potential bias, allowing for a reliable assessment of model performance.

4.0 Choice of the classifiers

Selecting appropriate classifiers is essential for building an effective predictive model, especially when dealing with imbalanced and complex datasets such as startup success prediction. In this project, three widely used classifiers were chosen: **K-Nearest Neighbors (KNN), Decision Tree, and Random Forest**.

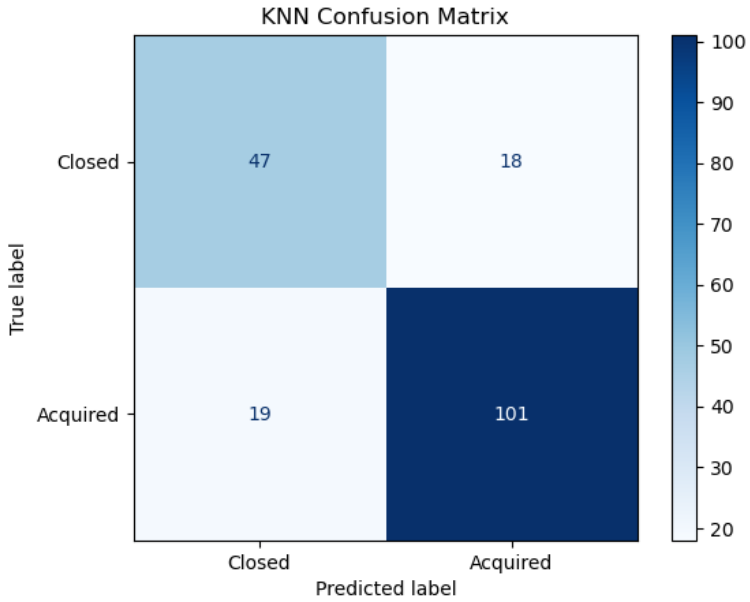
a. K-Nearest Neighbors (KNN)

Reason for Selection:

1. KNN is a **non-parametric, instance-based learning algorithm** that classifies data points based on the majority class among their closest neighbors.
2. It works well when the decision boundary is **complex and non-linear**, making it useful for capturing startup success trends.
3. KNN is simple to implement and interpretable, making it a great choice for comparison with tree-based models.

Impact on Performance:

1. Sensitive to **feature scaling**, requiring normalization for optimal performance.
2. Performance depends on the choice of **K (number of neighbors)**, which influences model bias vs. variance trade-off.
3. Since KNN does not support built-in class weighting, imbalance in the dataset can impact classification accuracy.



b. Decision Tree Classifier

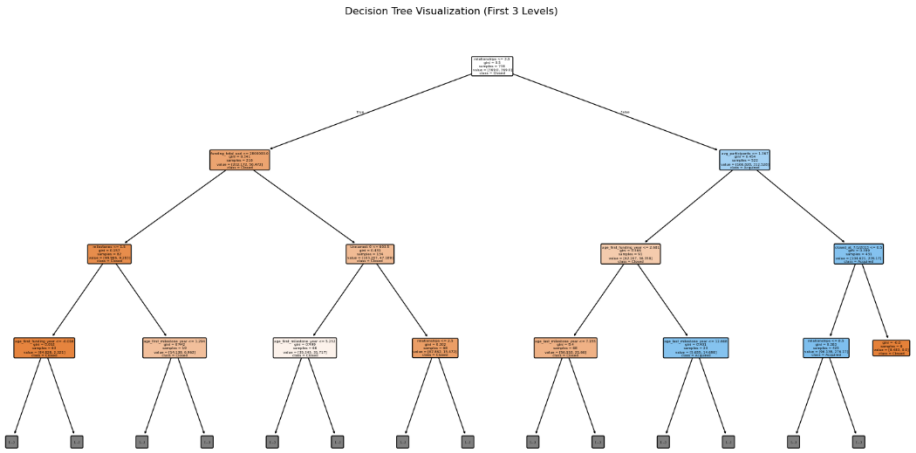
Reason for Selection:

1. Decision Trees provide **hierarchical decision-making**, allowing for easy interpretation of classification rules.
2. Unlike KNN, Decision Trees are less affected by feature scaling and can handle missing data well.

3. They are ideal for capturing **conditional relationships** in startup success prediction.

Impact on Performance:

1. Can suffer from **overfitting**, especially with deep trees, requiring tuning via **pruning**.
2. Provides **feature importance scores**, which help in identifying key predictors.
3. Works well with **class weighting**, making it useful for imbalanced datasets.



c. Random Forest Classifier

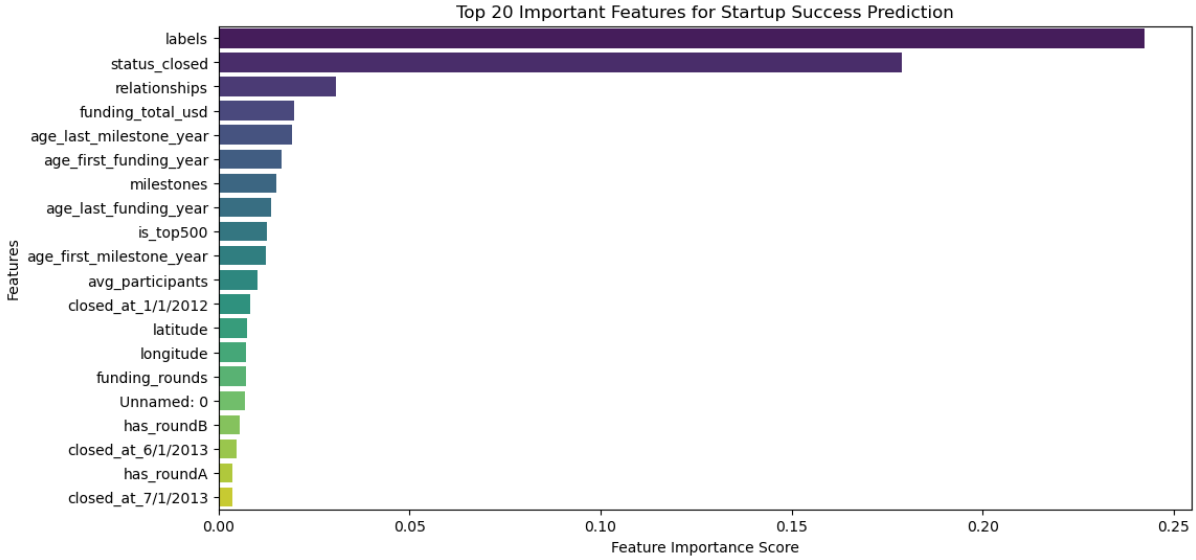
Reason for Selection:

1. Random Forest was primarily used to **determine feature importance**, helping identify the most influential predictors for startup success or failure.
2. Unlike single decision trees, Random Forest provides a more **robust feature ranking** by averaging across multiple trees, reducing bias toward individual features.
3. It helps in **dimensionality reduction**, ensuring that only the most relevant features are considered for modeling, improving efficiency and interpretability.
4. Additionally, Random Forest supports **handling class imbalance** through class weighting, making it useful for analyzing biased datasets.

Impact on Performance:

1. **Enhanced Feature Selection** → By analyzing feature importance scores, irrelevant or weak features can be removed, improving model accuracy.

2. **Better Model Interpretability** → Helps explain which factors (e.g., funding rounds, investor involvement) contribute most to predicting startup success.
3. **Optimized Computational Efficiency** → Selecting only high-impact features reduces complexity and speeds up training for other models like KNN and Decision Trees.
4. **Improved Generalization** → Removing unimportant features minimizes overfitting, making the final model more adaptable to unseen data.



d. Evaluation Metric

Given the **imbalanced nature** of the dataset, the **F1-score** was chosen as the primary evaluation metric. The F1-score balances **precision and recall**, making it more informative than accuracy for imbalanced classification problems.

In this project, **Hold-Out Validation** was used instead of cross-validation. The dataset was split into **training (80%) and testing (20%)** sets to evaluate model performance on unseen data. Since the dataset is imbalanced, **class weighting was applied to Decision Tree and Random Forest** to enhance prediction fairness. KNN, however, does not support class weighting, making it potentially more sensitive to class imbalance.

5.0 Conclusion

In this study, we analyzed the **startup success prediction dataset** and implemented **K-Nearest Neighbors (KNN), Decision Tree, and Random Forest** classifiers to assess predictive performance. Given the **imbalanced nature of the dataset**, where **597 startups were acquired (64.7%)** and **326 startups closed (35.3%)**, careful preprocessing was necessary to ensure fair learning outcomes.

To **mitigate class imbalance**, **class weighting** was applied to **Decision Tree and Random Forest**, allowing the models to correctly prioritize minority-class instances.

Meanwhile, **Random Forest was also utilized for feature importance analysis**, identifying the most influential predictors for startup success.

Model performance was evaluated using **hold-out validation**, with an **80-20 train-test split** to measure generalization ability on unseen data. The **F1-score** served as the primary evaluation metric, as it balances **precision and recall**, making it ideal for imbalanced classification tasks.

References

Baskoro, H., Prabowo, H., Meyliana, M., & Lumban Gaol, F. (2022). Predicting Startup Success, a Literature Review. *International Conference on Information Science and Technology Innovation (ICoSTEC)*, *1*(1), 123–129. https://doi.org/10.35842/icostec.v1i1.6

Lotfalia, M. H., Hashemian, S. M., & Akbari, M. (2022). Predicting the success of startups Using machine learning. *Journal of Innovation and Entrepreneurship*. https://doi.org/10.1186/s13731-024-00436-x

Nikhitha, V., Soumya, K., Reddy, A. R., Kumar, G. A., & Vani, M. S. (2025). *International Journal of Research Publication and Reviews Startup Success Rate Prediction Model*. *6*(4), 3668–3674.

Sharchilev, B., Ozornin, D., Roizner, M., Serdyukov, P., Rumyantsev, A., & De Rijke, M. (2018). Web-based startup success prediction. *International Conference on Information and Knowledge Management, Proceedings*, 2283–2292. https://doi.org/10.1145/3269206.3272011