Vegetation Data Classification Report

Name: Adam Yuflindra Patmawidjaja

Student ID: 19014854

1. Summary

The report includes the development of a classification model to predict the vegetation condition on Wadjuk Noongar Country area, a significant task to ensure satisfactory land management and environmental monitoring. The project leveraged a dataset with up to 5000 entries which included vegetation data which contain features surrounding the 3 different classes of a vegetation condition including healthy, degraded, or at risk. This data is to be used on the unseen test data which only contains up to 500 entries. Thus, the project evaluated four machine learning models, namely K-Nearest Neighbour (KNN), Decision Tree (DT), Naive Bayes (NB), and Logistic Regression (LR). The analysis found that two of the four models, namely DT and KNN models consistently achieved the best performance based on accuracy and f1-score. A comparative analysis, with and without Principal Component Analysis (PCA) for dimensionality reduction, further confirmed the strength and accuracy of these models and their potential for identifying key vegetation categories, including the rare (508/5000) yet critical “At Risk” attributes.

2. Methodology

2.1. Data Preparation

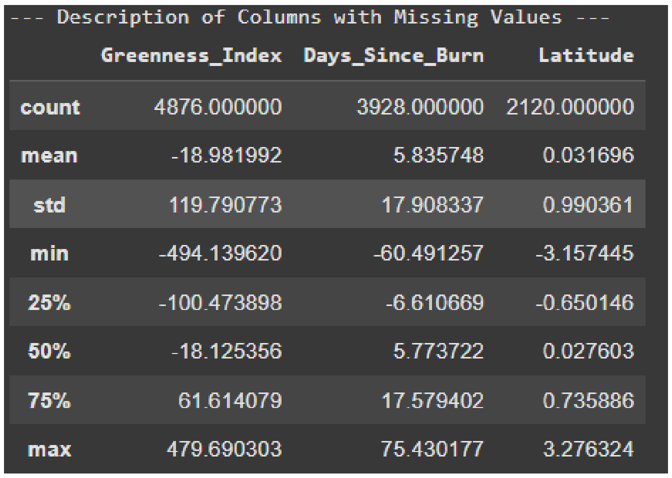
2.1.1 Data Collection and Loading

The vegetation health classification dataset was retrieved from an SQLite database containing environmental and vegetation measurements. The training dataset consisted of 5,000 entries with 32 features, including vegetation properties, water metrics, etc. This dataset include the following key characteristics:

- Dataframe dimensions: 5,000 rows x 32 columns
- Key Variables: ‘class’ with three categories (Healthy, Degraded, at Risk)
- Mix of numerical continuous (21 features), numerical discrete (5 features), and categorical (6 features)
- Index column present for record identification

2.1.2 EDA and Missing Value Analysis

Initial analysis revealed missing values in three important features:



--- Description of Columns with Missing Values ---

	Greenness_Index	Days_Since_Burn	Latitude
count	4876.000000	3928.000000	2120.000000
mean	-18.981992	5.835748	0.031696
std	119.790773	17.908337	0.990361
min	-494.139620	-60.491257	-3.157445
25%	-100.473898	-6.610669	-0.650146
50%	-18.125356	5.773722	0.027603
75%	61.614079	17.579402	0.735886
max	479.690303	75.430177	3.276324

The diagram shows no duplicate records, indicating good data quality in terms of uniqueness. Although, distribution analysis of features with missing values shows approximately normal distributions with some skewness and outliers, particularly in the Greenness_Index and Days_Since_Burn features.

2.1.3 Missing Value Imputation Strategy

A median imputation strategy was chosen over mean imputation for the following:

- **Robustness to Outliers:** The median provides the middle value when data is sorted, making it less sensitive to extreme value compared to the mean
- **Distribution Preservation:** Median imputation better maintains the original distribution shape, especially for data with slight skewness like in Greenness_Index
- **High Variance Handling:** Features like Days_Since_Burn (std = 17.91 days) and Greenness_Index (std = 119.79) showed high spread, where median would provide a more stable central tendency [1]

This imputation process is necessary to prevent any data leakage before preprocessing. The preprocessing function would potentially run an error message upon encountering the NaNs or transform the NaNs into zero without appropriate formulation, skewing the data further.

2.1.4 Feature Selection and Dimensionality Considerations

Latitude and Longitude coordinates were removed from the analysis based on the following points:

- **M Redundant Information:** Environmental factors influenced by location (elevation, soil type, climate) were already captured by features in the dataset, and with those addition it would just skew distributions more
- **Model Complexity Reduction:** Redundancy aside, removing these features enhances the model while focusing on directly measurable environmental indicators like SoilPh and Water Level.
- **Feature set optimisation:** Original features: 32 into 28 final features after the removal of index, latitude, longitude, and the targeted class data. Whereas, features retained are mainly vegetation health and properties.

2.1.5 Class Distribution Analysis and Imbalance Assessment

Target Variable Distribution (Training Data) include 2489 samples of healthy plants, 2003 samples of degraded plants and 508 samples of plants at risk. The analysis revealed some class imbalance, particularly with the AtRisk category representing only more or less 10% of samples. This distribution requires attention in model training to ensure adequate representation of minority classes.

This assessment would imply that there is some risk of model bias toward majority classes like Healthy and Degraded, potentially putting poorer performance on AtRisk vegetation identification. This highlights the importance of effective sampling methods for the model training.

2.1.6 Data Preprocessing Pipeline Development (Scaling)

Numerical Feature processing for up to 26 features:

- Standardisation: Applied the StandardScaler function to normalise feature scales.
- Range Normalisation: Ensured all numerical features contribute equally to distance-based algorithms

Categorical Feature Processing for up to 5 categorical features:

- One-Hot encoding applied to the five categorical features including Soil_Type, Burn_Season, Surface_Water_presence, Landform, and Human_Disturbance
- Drop-First Strategy prevented multicollinearity by removing a dummy variable per category
- Unknown Category Handling configured to handle the unseen categories in test data

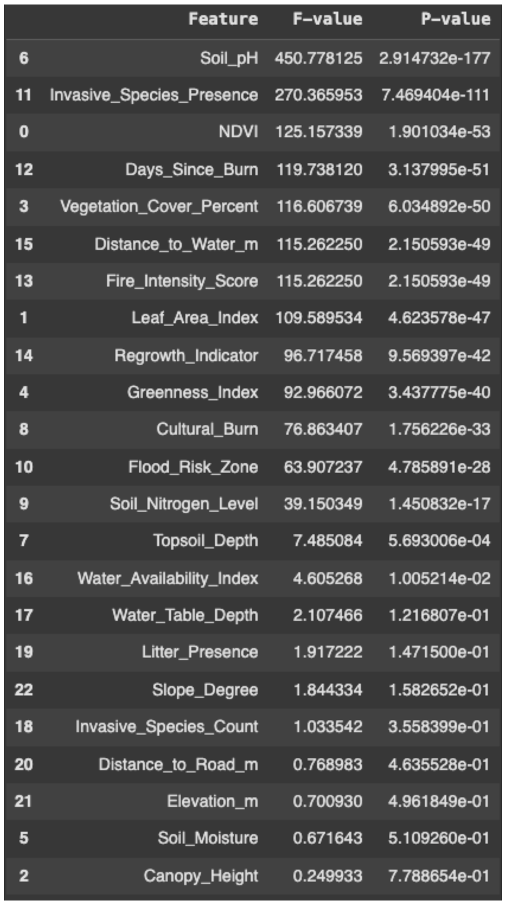
Pipeline Integration of both Numerical and Categorical features:

- Combined using ColumnTransformer
- Ensured consistent transformation across training and test datasets (applied later to unseen also)
- Maintained feature independence and prevented further data leakage

2.1.7 Data Preparation Impact Summary

The comprehensive data preparation process successfully transformed raw environmental measurements into a ML-ready dataset. The following are some quality improved and identified for further tuning:

- 100% missing value resolution through median imputation
- Class imbalances representation to be balanced via SMOTE
- Standardised feature scaling for algorithm compatibility
- Categorical feature encoding for numerical processing



	Feature	F-value	P-value
6	Soil_pH	450.778125	2.914732e-177
11	Invasive_Species_Presence	270.365953	7.469404e-111
0	NDVI	125.157339	1.901034e-53
12	Days_Since_Burn	119.738120	3.137995e-51
3	Vegetation_Cover_Percent	116.606739	6.034892e-50
15	Distance_to_Water_m	115.262250	2.150593e-49
13	Fire_Intensity_Score	115.262250	2.150593e-49
1	Leaf_Area_Index	109.589534	4.623578e-47
14	Regrowth_Indicator	96.717458	9.569397e-42
4	Greenness_Index	92.966072	3.437775e-40
8	Cultural_Burn	76.863407	1.756226e-33
10	Flood_Risk_Zone	63.907237	4.785891e-28
9	Soil_Nitrogen_Level	39.150349	1.450832e-17
7	Topsoil_Depth	7.485084	5.693006e-04
16	Water_Availability_Index	4.605268	1.005214e-02
17	Water_Table_Depth	2.107466	1.216807e-01
19	Litter_Presence	1.917222	1.471500e-01
22	Slope_Degree	1.844334	1.582652e-01
18	Invasive_Species_Count	1.033542	3.558399e-01
20	Distance_to_Road_m	0.768983	4.635528e-01
21	Elevation_m	0.700930	4.961849e-01
5	Soil_Moisture	0.671643	5.109260e-01
2	Canopy_Height	0.249933	7.788654e-01

Figure 1 – X_train features ANOVA f-test results

This quality is then tested with the ANOVA F-Test Feature Validation. The ANOVA F-test was performed to check the statistical relationship between each numerical feature (e.g.: Soil PH) against the targeted 'class'. The F-value is the key indicator of group separation in which a higher F-value suggests a much stronger more significant relationship with the target variable [2]. Whereas, the P-value indicates the statistical significance which means very low P-values allows us to reject the null hypothesis or feature does have a meaningful relationship with the vegetation class [2]. There are 15 significant features that shown robust relationship with p-value <0.05 such as Soil_pH (F-value: 450.78, P-value: 2.91e-177) and NDVI (F-value: 125.16, P-value: 1.90e-53). On the other hand, there are 8 values with weak relationships yet remain significant such as Canopy_height (F-value: 0.25, P-value: 0.78). This significance will be proven later with the Principal Component Analysis configuration comparison which shows the significance of withholding the full dimension instead of reduced.

2.2 Classification

2.2.1 Train-Test Split and Class Balancing

The splitting strategy revolved around the following:

- Split Ratio of 1:4, where 80% is training (4,000 samples) and 20% is testing (1,000) from the master training data.
- The Stratified KFold Sampling leveraged to maintain original class proportions in both splits
- Random State for fixed seed (42)

As previously mentioned, there are class imbalances in the data, providing a space for SMOTE application for class balancing. SMOTE or Synthetic Minority Oversampling Technique is leveraged to address the following class imbalance:

Before Smote:

- At Risk: 406 (10.15%)
- Degraded: 1,603 (40.08%)
- Healthy: 1,991 (49.78%)

After Smote:

- All classes balanced to 1,991 samples each (33.33%) for training
- Total training samples increased from 4,000 to 5,973

As the name suggests, SMOTE works by creating new, 'synthetic' examples from the minority class [3]. It does this by selecting a sample from the minority class, in this case 'AtRisk', finding its k-nearest neighbours, and then generating a new sample along the line segments connecting the chosen sample to its neighbours [3]. This process expands the feature space of the minority, allowing the model to learn its patterns easily without having to duplicate existing data [3].

Thus, it generates synthetic samples rather than just simple duplication, which significantly improves the model's overall sensitivity against those minority class patterns. Most importantly, it manages to achieve that while still holding the integrity of the original data, all while giving the minority class like 'AtRisk' a significant boost in representation. Shown below is a diagram that compares the difference of before and after SMOTE.

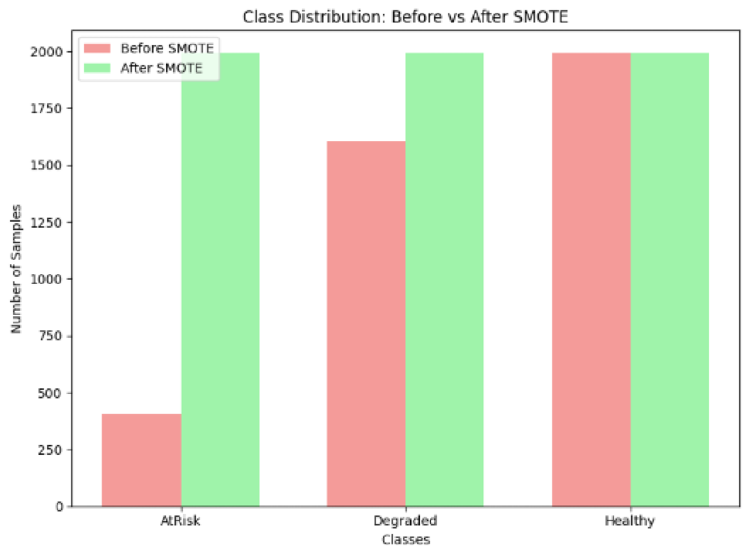


Figure 2 – SMOTE Before vs After Balancing Comparison

2.2.2 Cross-Validation Strategy Evaluation

We looked at three main ways to handle the cross-validation including: [4]

- **KFold:** Splitting the data sequentially
- **ShuffleSplit:** Leveraging random splits that can slightly overlap
- **StratifiedKFold:** Maintaining similarity in class distribution across all folds

Selected Approach: StratifiedKFold (shown at 10 splits) was chosen for hyperparameter tuning as it ensures each fold maintains the original class distribution, crucial for imbalanced datasets like this. The cross-validation method is shown in the following diagram:

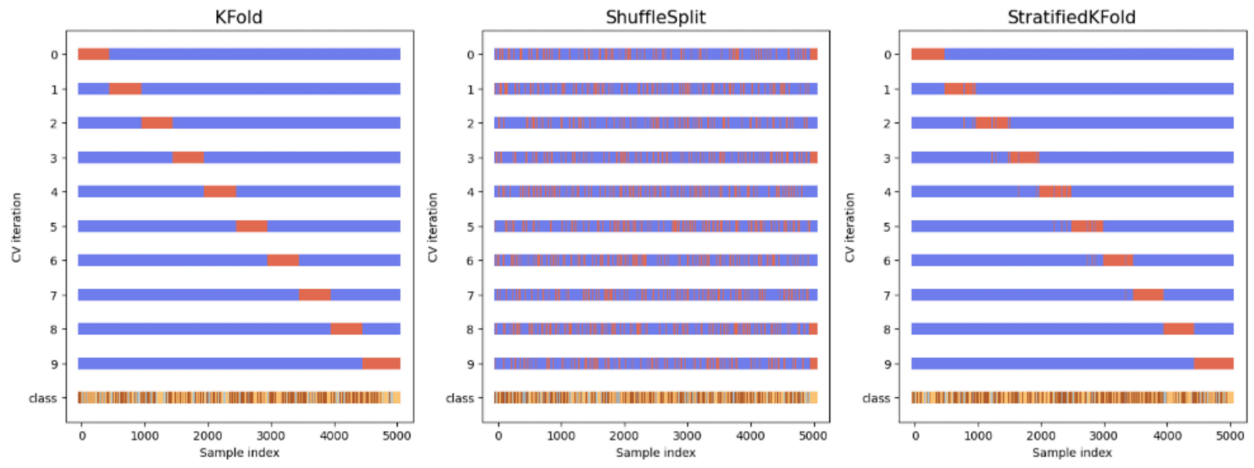


Figure 3 – Cross-validation Approach Comparison

Despite the StratifiedKFold sampling in figure 3 showing some gaps, the distributions are rather more evenly clustered than ShuffleSplit in which data mostly overlap. Whereas, KFold basically had not done any distribution but a stratification by pattern-based instance at random order, as long as they do not overlap. The gaps StratifiedKFold can be seen to not overlap but rather randomly sampled in another split, suggesting that these are some gaps which should not have been in the same split in non-StratifiedKFold. This Stratified model is arranged in a way that it fits the KFold pattern, whereas in actual workings it is randomised for fixed seed (42).

2.2.3 Model Selection Strategy and Rationale

This modelling opted for four diverse machine learning algorithms to the vegetation classification problem, each offering distinct advantages for different aspects of the dataset:

Algorithm	Rationale	Advantages	Disadvantages
K-Nearest Neighbors (KNN)	Selected for its simplicity and expected effectiveness with environmental data, as similar vegetation conditions should result in similar classifications.	No assumptions about data distribution, handles non-linear relationships naturally, and is robust to outliers when using appropriate distance metrics.	Sensitive to feature scaling and dimensionality, and can be computationally expensive for large datasets.
Decision Tree	Chosen for its interpretability and ability to capture complex feature interactions, revealing critical environmental thresholds.	Highly interpretable, handles both numerical and categorical features, captures non-linear relationships and feature interactions.	Prone to overfitting, and is sensitive to small changes in the data.
Naive Bayes (Gaussian)	Included as a probabilistic baseline model for fast training and prediction, despite its simplifying assumption of feature independence.	Fast training and prediction, works well with small datasets, and handles multiple classes naturally.	Its strong independence assumption may not hold true for complex environmental features.
Logistic Regression	Selected as a linear baseline model to provide a benchmark for more complex models and to reveal linear feature relationships.	Provides probability estimates, is less prone to overfitting, and is computationally efficient.	Assumes linear relationships, meaning it may struggle with complex feature interactions.

Model Complexity: The strategy balanced the use of simple linear models like LR and NB with more complex, statistical approaches like KNN and DT as for a scholarly paper, understanding feature importance is crucial for providing meaningful yet sophisticated ecological insights.

2.2.4 Hyperparameter Tuning Strategy

Hyperparameter grids were designed for each of the four algorithms to ensure optimal performance while keeping the computational load manageable, considering that the coding is conducted over cloud and not directly under CPU performance. The selection of these ranges was based on both established best practices and preliminary experiments. The process utilised various AI modelling tools including Julius, Claude, Bolt, and Gemini and overall the following are the agreed main purpose behind this hyperparameter tuning:

- **KNN:** The hyperparameters including n_neighbours of [3, 5, 7, 9 ,11] or odd numbers to prevent ties in voting, balancing local versus global decision boundaries. The weights are set to [‘uniform’, ‘distance’] and metrics were set to [‘euclidean’, ‘manhattan’]to test different ways of calculating neighbour contributions
- **Decision Tree:** The Max depth was set to [3, 5, 7, 10, None] to prevent both underfitting (too shallow) and overfitting (too deep), while still maintaining model interpretability. The min_samples_split and min_samples_leaf were set to [2, 5, 10] and [1, 2, 4] respectively for decision tree model integrity during training per split. Additionally, the criterion is set to [‘gini’, ‘entropy’] to compare impurity measures for splitting, also

attributing towards the integrity

- **Naive Bayes:** Simply setting a Laplace Smoothing by setting var_smoothing to [1e-9, 1e-8, 1e-7, 1e-6] or in a tight range to ensure numerical stability during calculations
- **Logistic Regression:** Setting up the regularisation parameters with C: [0.01, 0.1, 1, 10, 100] with penalty: [‘11’, ‘12’], basically chosen on a logarithmic scale to cover the spectrum from weak to strong regularisation.\

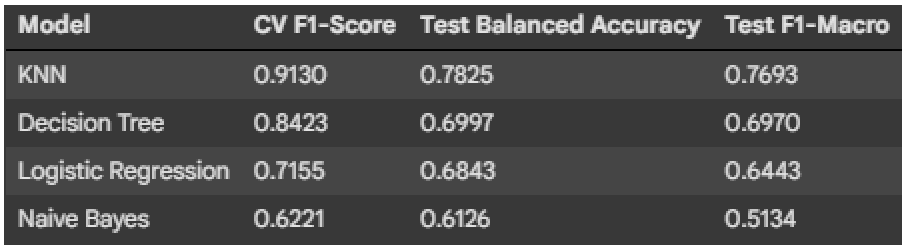
2.2.5 Training Implementation Details

The nuts and bolts surrounding this training revolves around the three main factors:

- **Computational Configuration:** To ensure that the training was as quick as possible, n_jobs=-1 on GridSearchCV was utilised for the purpose of multi-core processing. Even though this programming was done via cloud, this would improve the performance if it were to be conducted via a self-run Jupyter Notebook.
- **Reproducibility:** Pipelines are designed in a way that minimise memory usage through efficient data transformations like the utilisations of pre-defined functions and step by step random state processing across every random component from train_test_split to SMOTE and the cross-validation. This ensures that anyone running this code will get the exact same results
- **Training Monitoring:** Cross-validation scores tracked, best parameter identification are defined on performance including F1-macro score and Balanced Accuracy, and final model evaluation is only on the held out test set

2.2.6 Model Training Results

The training process simultaneously trained all 4 classifier models. While it took some time to load and complete the training considering the size and dimensionality of the dataset, the following results were extensively provided in much detail:



Model	CV F1-Score	Test Balanced Accuracy	Test F1-Macro
KNN	0.9130	0.7825	0.7693
Decision Tree	0.8423	0.6997	0.6970
Logistic Regression	0.7155	0.6843	0.6443
Naive Bayes	0.6221	0.6126	0.5134

Figure 4 – Model Performances

By order, the first model training was conducted on the split test set (1000 samples split) with the KNN model from the SMOTE balancing and preprocessors after the split. The model demonstrated superior overall performance compared to other models and was effective at handling all classes in the dataset with moderate precision (48%) in ⅓ of the classes. Shown the following diagram are the products of the model with **78.25% balanced accuracy** and an **F1-Macro of 76.93%**:

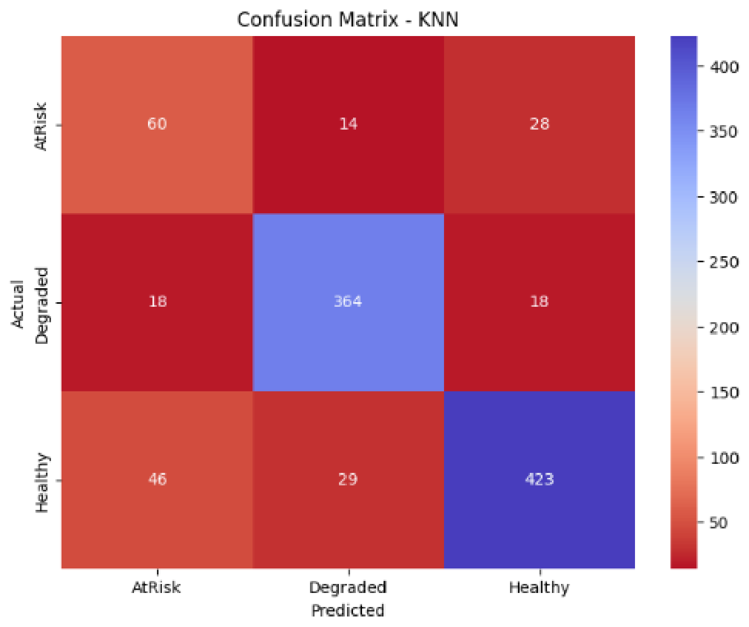


*Figure 5 – Confusion Matrix shows a great model performance of KNN with **Best Parameter Identified as 'Manhattan' From Weights of 'Distance' and n_neighbours of** 7*

Thereafter, the second model training of Decision Tree was used to train the split test set (1000 sample split) based on the previously tuned hyperparameters. Although, unlike the KNN, it demonstrated good interpretability and a balanced performance across all classes at best with a significantly lower precision (41%) in ⅓ of the classes. Shown in the following diagram are the decision trees with depth maxed at 3 and unlimited to for readability and estimations under a barely 70% score both for accuracy (69.97%) and F1-Macro (69.7%):

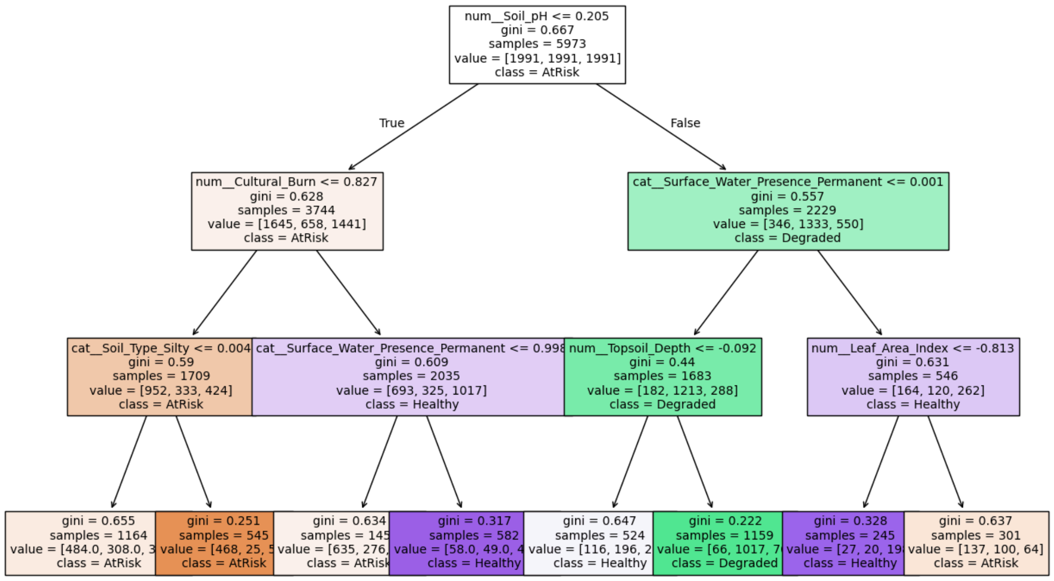


Figure 6 – Classification Test Set Decision Tree with Max Depth = 3 from Split Test Samples

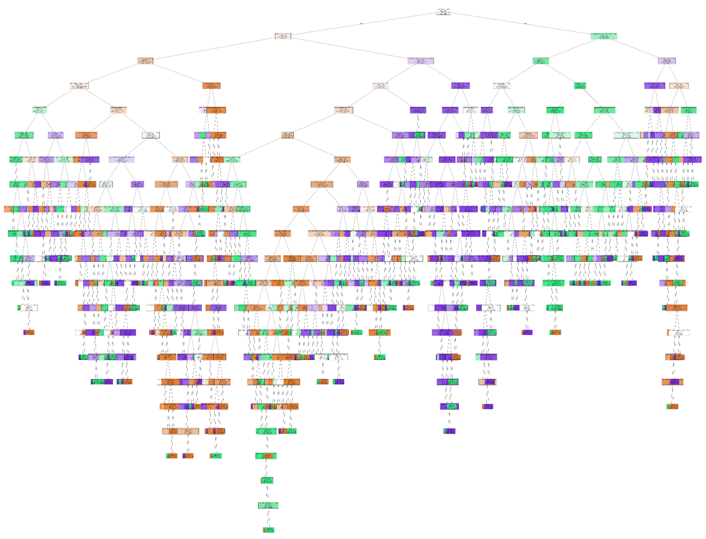


Figure 7 – Classification Test Set Decision Tree with Unlimited Depth has >15 root nodes and split points towards the leaf nodes

Shown in Figure 8 is the confusion matrix of the DT model which suggests that while it has less false positives in the 'AtRisk' class, it actually made a significantly higher false positive result in 'Degraded' class which resulted to a lower precision '41%' precision in ⅓ of the classes.

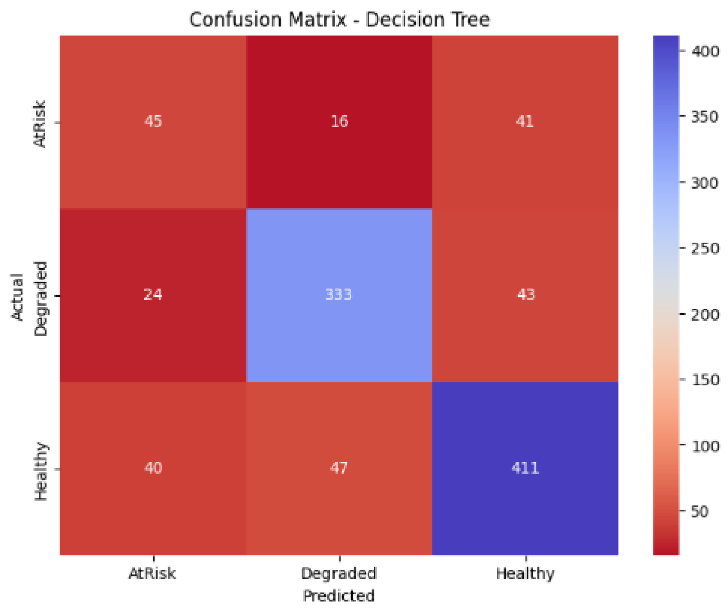
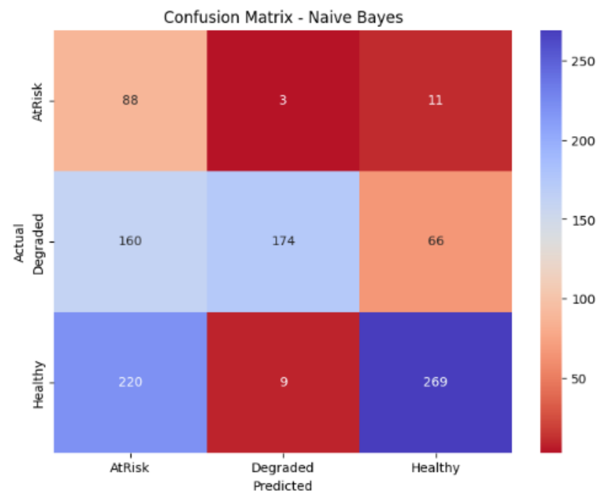


Figure 8 – The Confusion Matrix for the Decision Tree Model from Split Test Samples with ***Entropy classifier From no max_depth, 2 minimum sample leaf, and sample split of 10***

Followed by the other two linear models, namely NB and LR. The two models represent the Linear Regression method of data classification which assumes the standard of linearity and probability. LR had a close proximity in terms of balanced accuracy (**68.43%**) against DT, however falls off upon F1-macro scoring (**64.43%**) due to the complex nature of the vegetation data. This F1-macro limitation is also held by NB considering its >10% distance (**51.34%**) of F1-macro score against the 2nd best model, DT, while also holding a significantly **>3%** lower accuracy (**61.26%**) against LR. The two models shared the same limitation of failing to account for data complexity. The products can be shown in the following confusion matrixes:



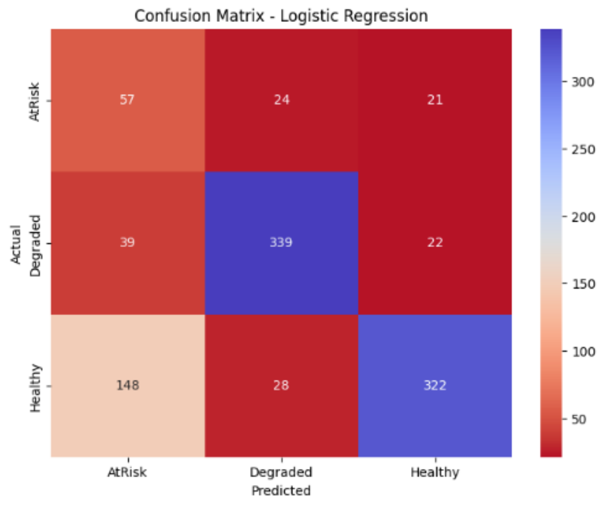


Figure 9 – The Confusion Matrix Comparison of the two Linear Models, NB and LR.

2.2.7 PCA Comparison

This dataset contains up to 5,000 entries with over 30 different features which are proven to be significant for the classification of vegetations. Considering the dimensionality, it would be wise to consider a possible reduction to improve the models and hasten the training process. A quarter of the dimensions are shown below in N-D space plots:

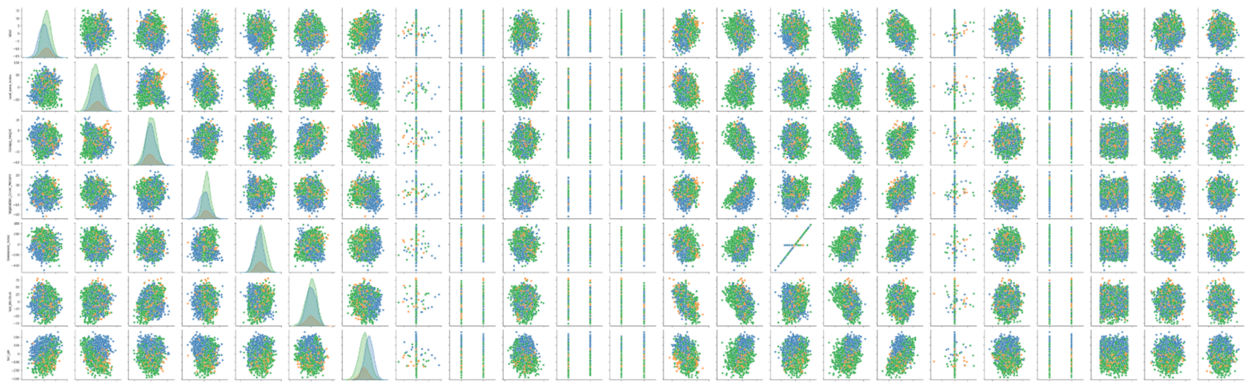


Figure 10 – N-D Space Plots Parameters

The PCA method allows a dimensionality reduction without the need to further eliminate any features, for the purpose of enhancing the modelling process [5]. Although, another pipeline must be created to separate the default data from the PCA preprocessed data. The two are separated as we must not assume that it would improve the process straightaway. The pipeline will allow the following reduction, shown by the dashes:

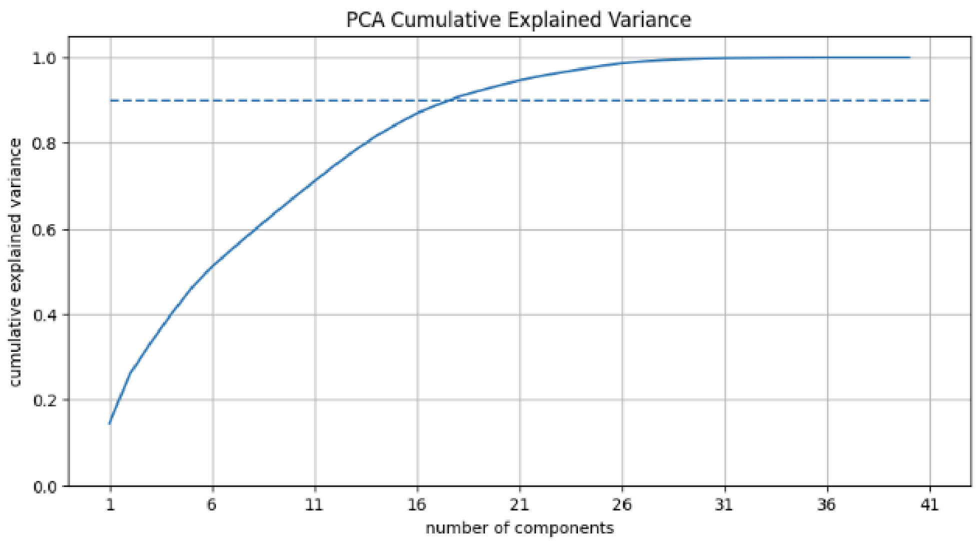
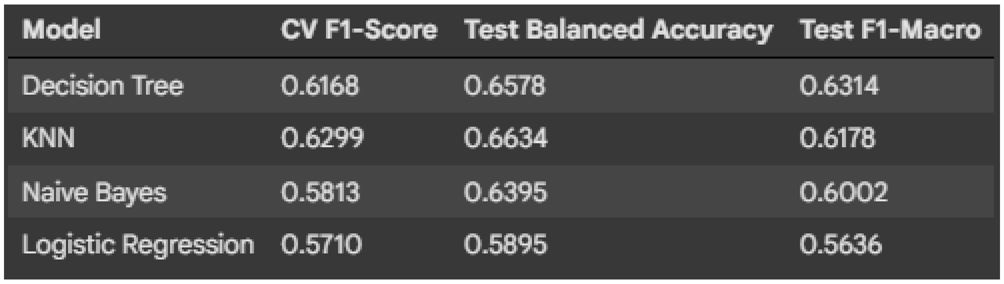


Figure 11 – Dimensional Reduction Line Graph

Data will be saved as X_train_pca and Y_test_pca to carry out a separate model training. However, the result below has shown that a dimensionality reduction may not always be the case with complex data like vegetation:



Model	CV F1-Score	Test Balanced Accuracy	Test F1-Macro
Decision Tree	0.6168	0.6578	0.6314
KNN	0.6299	0.6634	0.6178
Naive Bayes	0.5813	0.6395	0.6002
Logistic Regression	0.5710	0.5895	0.5636

Figure 12 – Classification Results on 4 Models with PCA Preprocessed Data

Shown in Figure 12, a dimensional reduction had instead caused significant performance degradations:

- **KNN:** 15.4% decrease in balanced accuracy, 19.7% decrease in F1-Macro
- **Decision Tree:** 6% decrease in balanced accuracy, 9.4% decrease in F-1 Macro
- **Naive Bayes:** 5% decrease in balanced accuracy, 4.4% decrease in F-1 Macro
- **Logistic Regression:** 4.5% decrease in balanced accuracy, 5% increase in F-1 Macro
- **Consistent Pattern:** All models **performed better on default preprocessed data, and both KNN and DT are also the best performers under PCA.**

Potential reasons for PCA Underperformance:

- **Linear Transformation Loss:** Its linear approach lost crucial non-linear relationships needed to distinguish the vegetation classes, in particular 'AtRisk' class. This is a similar problem with the linear models like NB and LR
- **Minority Class Mixture:** It prioritised overall variance, effectively mixing the specific feature combinations important for accurately identifying the underrepresented 'AtRisk'

2.2.8 Key Findings and Deployment

From the accuracy of the three models and comparison with the PCA data:

Primary Model: Deploy K-Nearest Neighbours with the following specifications:

- Parameters: n_neighbours=3, weights='distance', metric='manhattan'
- Preprocessing: StandardScaler, one-hot encoding, and median imputation
- Class Balancing: SMOTE with fixed random_state (42) for consistency

Secondary Model: Maintain Decision Tree

- Parameters: 'entropy' criterion, no max depth, and 1 min_sample_leaf
- Usage: Decision rule extraction and KNN prediction validation

All preliminary steps of the training set are done, thus the classification for unseen data can proceed. The data will have to be called out again from 'Assignment2025S2.sqlite' as selected_test_data_df. Lastly, once all training is done, everything will be saved based on the CSV prerequisites and under the file name Answers_[studentid].csv

3. Conclusion

Finally, this project ends in the development and validation of a classification model to predict vegetation condition (healthy, degraded, at risk) across the Wadjuk Noongar Country area. This will provide a critical machine learning tool for enhanced land management and environmental monitoring as suggested in the report's summary.

This methodology of ML modelling relied on an extensive and complicated data preparation pipeline. This included the necessary application of median imputation to replace missing values strategically and resolving the significant class imbalance with the advanced SMOTE tool, particularly concerning the underrepresented 'At Risk' category which had put a lot of models at risk of dysfunctional. An important finding that approved the approach was the underperformance of PCA; while it slightly improved the performance of the models, it caused a substantial performance degradation (up to a 19.7% drop in F1-macro for the KNN model). This confirmed that the full feature set contained essential, non-linear discriminative information that the linear PCA transformation failed to preserve.

These non-linearity repercussions are also represented in the classifier model training as of the four ML algorithms evaluated, the K-Nearest Neighbour model demonstrated superior predictive capability. Tuned with optimal parameters like Manhattan and distance metric, the model was selected as the primary deployment candidate, achieving a final test **F1-Macro score of 76.93%** and a B**alanced Accuracy of 78.25%**. The Decision Tree model will be retained as a secondary resource for decision rule extraction and prediction validation, narrowly achieving a final test score of 70% in both **accuracy (69.97%)** and **F1-Macro (69.7%)**.

Overall, the final, fully-built pipeline is prepared to deliver accurate and reliable classification predictions on the unseen test dataset. This outcome provides valuable, validated support for ecological decision-making like vegetation and land-management in the Wadjuk Noongar Country region.

4. References

[1] Australian Bureau of Statistics. "Measures of Central Tendency." Australian Bureau of Statistics. 2023. Last modified February 2, retrieved from: https://www.abs.gov.au/statistics/understanding-statistics/statistical-terms-and-concepts/measures-central-tendency.

[2] Brownlee, Jason. 2020. "Statistical Hypothesis Tests for Machine Learning." Machine Learning Mastery. 2020. Last modified February 2, retrieved from: https://machinelearningmastery.com/statistical-hypothesis-tests-for-machine-learning/.

[3] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. “SMOTE: Synthetic Minority Over-sampling Technique.” *Journal of Artificial Intelligence Research* 16: 321–357. https://doi.org/10.1613/jair.953.

[4] scikit-learn developers. "User Guide: 3.1.2. Cross-validation iterators." *scikit-learn*. Accessed September 26, 2025. https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators.

[5] scikit-learn developers. "User Guide: 6.1. Principal component analysis (PCA)." *scikit-learn*. Accessed September 26, 2025. https://scikit-learn.org/stable/modules/decomposition.html#principal-component-analysis-pca.

[6] Use of AI acknowledgement is provided in the Coding Notebook.