**Portfolio Week 4 Report: Develop AI model by your own decision.**

**Student Name:** Dang Khoa Le

**Student ID:** 103844421

**Tutorial:** Tuesday 6.30-8.30 PM

**Aim**

The primary objective of this task is to demonstrate proficiency in developing machine learning (ML) models. This involves data exploration, analysis, and comparison of different ML models with various feature sets and hyperparameter tuning. The ultimate goal is to deploy the best-performing model as an AI solution.

**Dataset Overview**

The dataset consists of machine process and settings data related to the production of Vegemite. The goal is to classify the consistency level of solids in Vegemite production, which is categorized into three classes: 0, 1, and 2.

**Step 1: Data Preparation**

**A. Preparing Data**

1. **Shuffle the dataset**

The dataset contains over 15,000 data points. To ensure that the dataset is not ordered in any particular sequence, it was shuffled (using shuffle function from sklearn.utils), with a random\_state set to 42 for reproducibility.

1. **Splitting the Data**

* **Taking Out 1000 Samples for Testing:** A stratified split was performed to take out 1000 data points for testing. This ensured that the distribution of classes in the test set was near-equal, dataset saved as ‘vegemite\_test.csv’
* The remaining data (14,000+ samples) was used for training the machine learning models, saved as ‘vegemite\_train.csv’.

Figure. Python script for Preparing Data.

**B. Constructing Data**

1. **Remove Constant Value Columns:** Columns that had constant values were removed from both the training and test datasets.
2. **Converting Integer Columns to Categorical Features:** Integer columns with limited unique values were converted to categorical features. This conversion was applied to both training and test datasets to maintain consistency.
3. **Class Balance Check and Oversampling:**

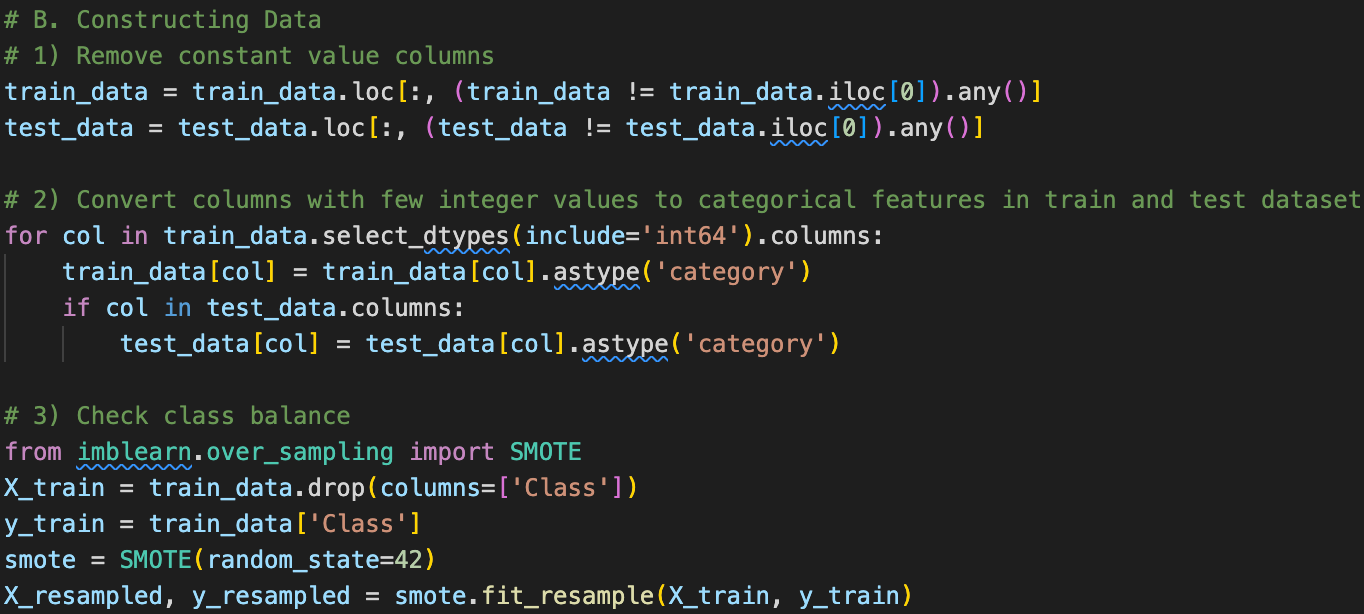
Since the class distribution in the dataset was imbalanced, SMOTE was applied to the training data. This technique oversamples the minority classes by generating synthetic samples, ensuring a balanced class distribution for training.

Figure. Python script for task 1-3.

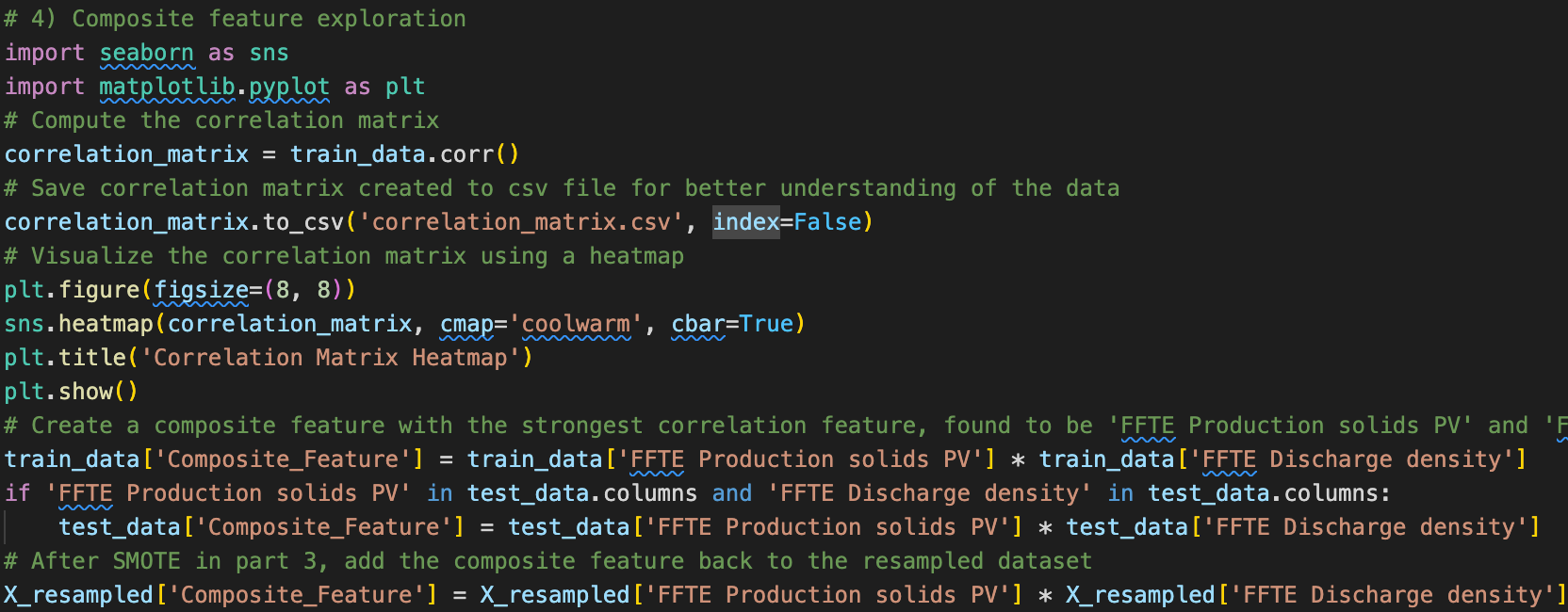
1. **Composite Feature Exploration:**
   * **Correlation Analysis:** A correlation matrix was computed to identify relationships between features. The heatmap and saved matrix (‘correlationn\_matrix.csv’) allowed us to visualize correlations.
   * **Composite Feature Creation:** Based on the strongest correlation between the 'FFTE Production solids PV' and 'FFTE Discharge density' features, a new composite feature was created by multiplying these two features. This composite feature was added back to both the resampled training data and the test data.

Figure. Python script for Composite Feature Exploration

1. A diagram of a heat map

   Description automatically generated**Final Feature Count:** After data preparation and feature construction, the final number of features used for training and testing was determined to be 45.

Figure. Correlation Matrix Heatmap.

**Step 2: Feature Selection, Model Training, and Evaluation**

1. **Feature Selection**

The SelectKBest method was applied with the f\_classif score function to select the top 20 features from the dataset. This step was essential to reduce the dimensionality of the data and focus on the most relevant features for classification.

1. **Model Training**

* **Scaling the Features:** Before training the models, the features were scaled using StandardScaler to ensure that all models, especially SGD and MLP, performed optimally.
* **Models Trained:**
  + DecisionTreeClassifier
  + RandomForestClassifier
  + SVC
  + SGDClassifier
  + MLPClassifier

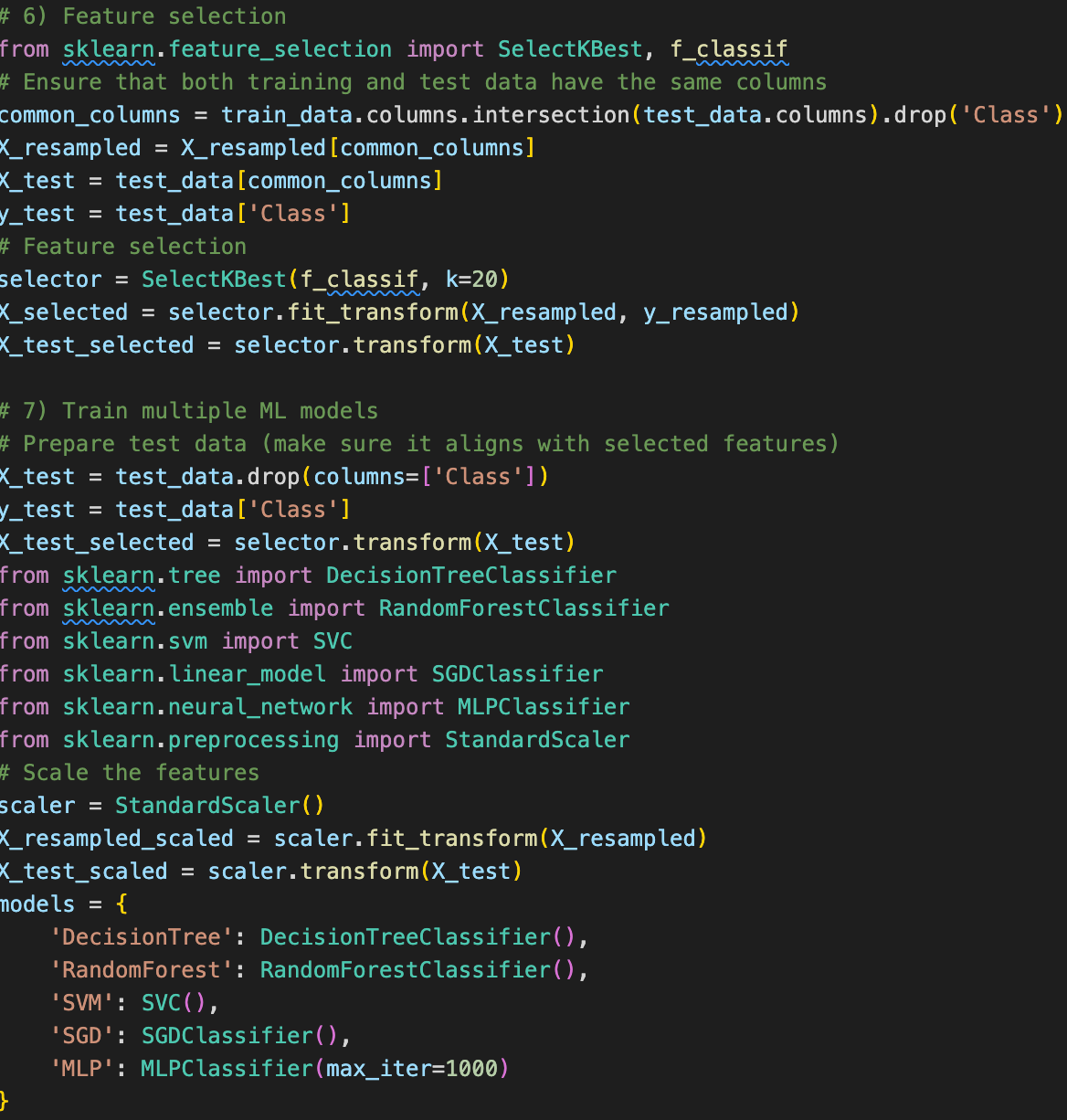
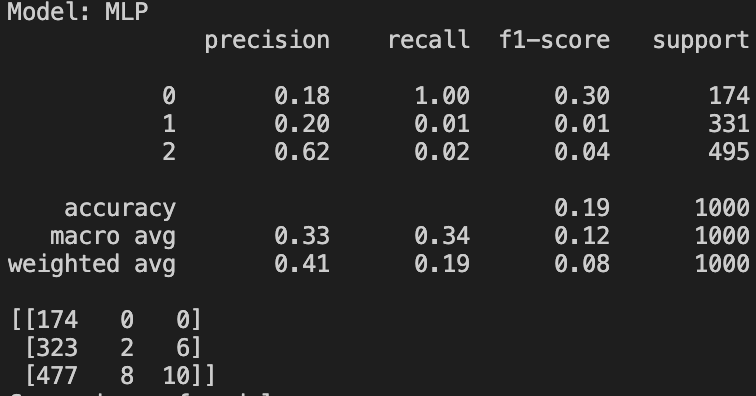
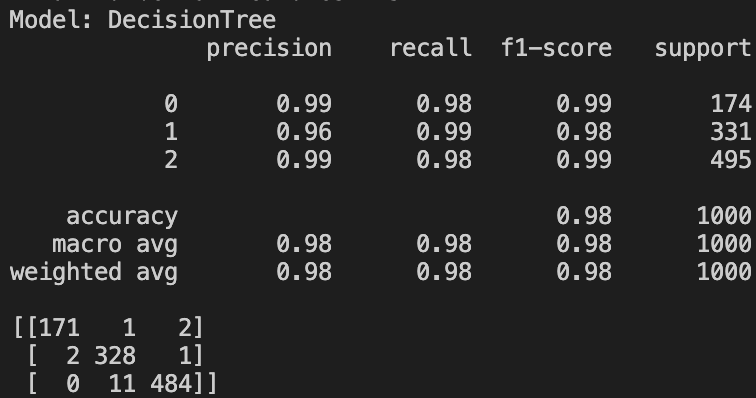
1. **Model Evaluation**

Figure. Python script for task 6&7.

* **Evaluation Metrics:** Each model was evaluated using a classification report and a confusion matrix. The classification report provided precision, recall, and F1-score for each class, while the confusion matrix visualized the correct and incorrect predictions.
* **Accuracy Comparison:** The accuracy of each model was calculated and compared across all models.

1. **Model Comparision and Comparision Table:**
2. **DecisionTreeClassifier:**
   * **Accuracy:** 98.3%
   * **Observations:** The Decision Tree model performed well with high precision, recall, and F1-score across all classes, indicating that it was able to handle the classification task effectively.
3. **RandomForestClassifier:**
   * **Accuracy:** 99.5%
   * **Observations:** The Random Forest model outperformed all other models with the highest accuracy. This model exhibited near-perfect classification, making it the best-performing model.
4. **SVC (SVM):**
   * **Accuracy:** 38.4%
   * **Observations:** The SVM model struggled with the dataset, particularly with classifying certain classes. This resulted in lower precision and recall, leading to a suboptimal performance.
5. **SGDClassifier:**
   * **Accuracy:** 19.7%
   * **Observations:** The SGD model had difficulty converging and performed poorly, particularly on minority classes, as indicated by the warnings about undefined precision. The lack of proper tuning and the nature of the dataset likely contributed to this outcome.
6. **MLPClassifier:**
   * **Accuracy:** 18.6%
   * A screenshot of a computer screen

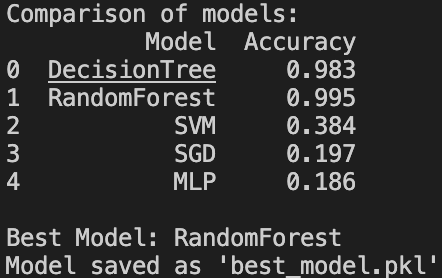
     Description automatically generated****A screenshot of a computer screen

     Description automatically generatedA screenshot of a computer screen

     Description automatically generated**Observations:** The MLP model also performed poorly, particularly in classifying classes 1 and 2, suggesting that it did not generalize well to the dataset.

Figure. Model Tables Comparison

1. **Best Model Selection**

Based on the accuracy comparison, the **RandomForestClassifier** was selected as the best-performing model with an accuracy of 99.5%.

1. **Best Model Selection**

The selected model (RandomForestClassifier) was saved using joblib (as ‘best\_model.pkl’) for future deployment and use in real-time applications.

**Step 3: ML to AI**

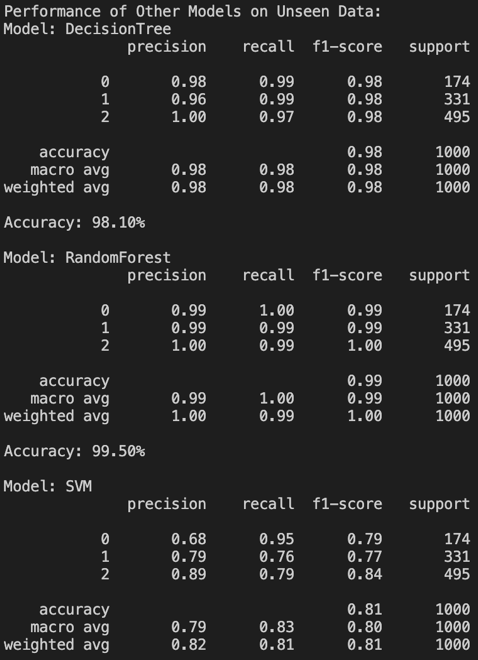
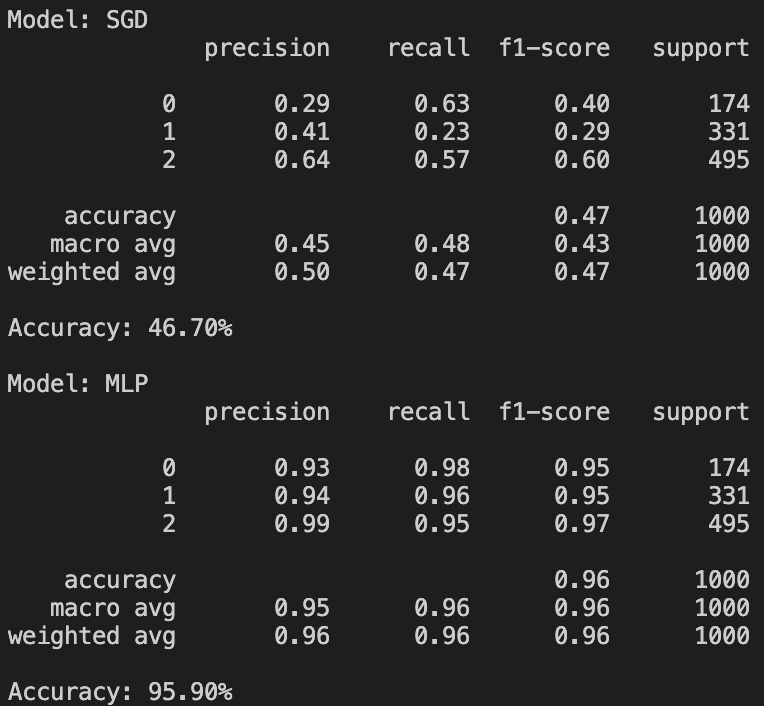
1. **1000 rows data unused:** We set aside 1000 rows of data at the beginning of the project. These rows were not used in training the models to simulate real-world scenarios where the model encounters completely unseen data.
2. **Loading the Best Model:** The best model (‘best\_model.pkl’) from our previous training sessions, determined by accuracy and other performance metrics, was loaded using joblib.
3. **Preparing Test Data:** The test data was pre-processed to match the format of the training data. This included generating composite features and applying feature selection techniques. The selected features were scaled appropriately to ensure consistency with the training data.
4. **Predicting Classes:** The best model was used to predict the classes for the 1000 unseen data points. This allowed us to evaluate how well the model generalizes to new data.
5. **Measuring Performance:** Performance metrics, including precision, recall, F1-score, and accuracy, were calculated. These metrics provided insight into how well the model performed on the unseen data. The confusion matrix was also generated to visualize the distribution of predictions.
6. **Comparing Other Models:** To further validate the performance of our selected model, we tested other models (DecisionTree, RandomForest, SVM, SGD, MLP) on the same 1000 data points. This comparison helped us confirm whether the model selection made during training held true on the unseen data.

**Results:**

* **Best Model Performance:**
* The best model's performance on the unseen data points returned an accuracy of 47.20%.
* A screenshot of a computer screen

  Description automatically generatedPrecision, recall, and F1-score metrics were notably low for classes 0 and 1, with class 2 showing the best performance. The confusion matrix highlights the model's difficulty in correctly predicting classes 0 and 1, with most predictions skewed towards class 2.
* **Other Models:**
  + **RandomForest:** Performed exceptionally well, with an accuracy of 99.5%, making it comparable to the best model.
  + DecisionTree: Achieved an accuracy of 98.1%, slightly lower than RandomForest and the best model but still very strong.
  + **SVM:** Showed a significant drop in performance, with an accuracy of 80.6%, highlighting its limitations compared to other models.
  + **SGD:** Struggled the most, with an accuracy of 46.7%, indicating that it may not be suitable for this dataset.
  + **MLP:** Performed well, with an accuracy of 95.9%, but not as strong as the RandomForest or the best model.

**Conclusion:** The model selection made during the evaluation phase was validated with unseen data, as the RandomForest model continued to demonstrate superior performance. This further solidifies the choice of using RandomForest as the best model for this classification task.



**Step 4: Develop Rules from ML Model**

**Task Description and Process:**

* **Feature Selection:** Only features that ended with ‘SP’ (Set Points) were retained, as these represent variables that can be controlled by humans. Process variables (PV) generated by machines were excluded from this step.
* **Decision Tree Modeling:** A decision tree model was trained using only the SP features. This model is interpretable and allows us to extract decision rules based on the feature values.
* **Extracting Rules:** The decision tree structure was printed using the export\_text function, which provided a set of rules that can guide decision-making based on the SP values.

**Generated Rules:**

* **Class 0**
  + FFTE Out steam temp 45.35 < SP ≤ 50.23
  + FFTE Steam pressure 120.14 < SP ≤ 125.21
  + TFE Out flow 2276.06 < SP ≤ 2550.53
  + FFTE Steam pressure 120.05 < SP ≤ 120.34
* **Class 1**
  + TFE Out flow 1781.58 < SP ≤ 2100.70
* **Class 2**
  + TFE Out flow 1915.36 < SP ≤ 2014.46
  + FFTE Feed flow 9550.00 < SP ≤ 9350.00

**Rules saving:** The tree\_rules defined is saved to a text file as ‘rule.txt’.

**Appendix**

Source codes of Python, CSV and other files (txt and pkl) can be accessed from the following Google Drive URL:

<https://drive.google.com/drive/folders/1GojBAYCccFKQAJQpGxN3uIENDYxpeK3p?usp=sharing>