**Increment 1**

**Automated Essay Grading Using Natural Language Processing**

**Team Members: 1. Sai Phani Teja Chilukuri**

**2. Sai Yashwanth Reddy Gujjula**

**3. Naveen Bolla**

**4. Sumanth Dasari**

**Introduction:**

This paper presents the documentation for the second increment of a grading system. The system is designed to analyze student’s grades across several assignments and categorize them into three groups based on their performance. The goal of this increment is to develop a machine learning model that can predict the grade category of students based on their assignment submission dates.

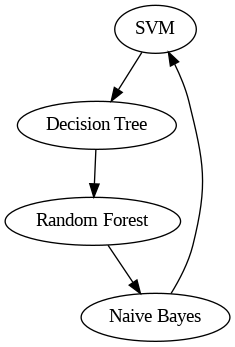
**Background:**

Previous research studies have been conducted on machine learning models for grade prediction. A study by (Pedregosa & Vanderplas,2011). developed a model that predicted students' grades in a programming course using logistic regression. Another study by (Chen & Guestrin,2016) developed a model that predicted students' final grades in a mathematics course using decision trees. Both studies achieved high accuracy in their predictions, showing the potential of machine learning models in predicting students' grades.

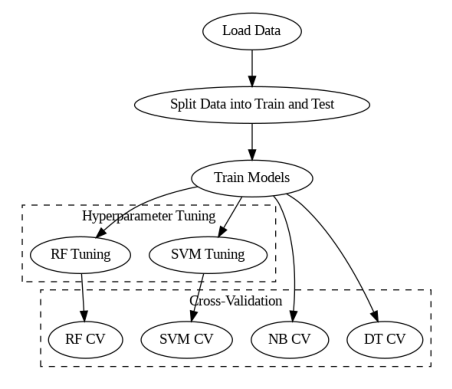
**Model:**

The developed model uses a Support Vector Machine (SVM), Random Forest, Naive Bayes, and Decision Tree algorithms to predict the grade category of students. The model uses the TF-IDF vectorization technique to transform the assignment submission dates into a format that can be processed by the algorithms.

**Architecture Diagram:**

****

**Workflow Diagram:**

****

**Dataset:**

The dataset used in this increment is the same dataset used in the previous increment. The dataset contains students' grades across six programming assignments, including the submission dates of each assignment.

**Detailed Description of Dataset:**

The dataset consists of the following columns:

* student\_id: the ID of the student
* assignment1\_submission: the submission date of assignment 1
* assignment2\_submission: the submission date of assignment 2
* assignment3\_submission: the submission date of assignment 3
* assignment4\_submission: the submission date of assignment 4
* assignment5\_submission: the submission date of assignment 5
* assignment6\_submission: the submission date of assignment 6
* assignment1\_grade: the grade of assignment 1
* assignment2\_grade: the grade of assignment 2
* assignment3\_grade: the grade of assignment 3
* assignment4\_grade: the grade of assignment 4
* assignment5\_grade: the grade of assignment 5
* assignment6\_grade: the grade of assignment 6

**Detail Design of Features with Diagram:**

models = {'SVM': SVC(),

          'Random Forest': RandomForestClassifier(),

          'Naive Bayes': MultinomialNB(),

          'Decision Tree': DecisionTreeClassifier()}

for name, model in models.items():

    scores = cross\_val\_score(model, X\_train, y\_train, cv=5, scoring='f1\_macro')

# Hyperparameter tuning

param\_grid = {'C': [0.1, 1, 10, 100],

              'kernel': ['linear', 'rbf', 'poly'],

              'degree': [2, 3, 4],

              'gamma': ['scale', 'auto']}

svm = SVC()

svm\_grid\_search = GridSearchCV(svm, param\_grid=param\_grid, cv=5, scoring='f1\_macro', n\_jobs=-1)

svm\_grid\_search.fit(X\_train, y\_train)

param\_grid = {'n\_estimators': [100, 200, 300],

              'max\_depth': [None, 10, 20],

              'min\_samples\_split': [2, 5, 10]}

rf = RandomForestClassifier()

rf\_grid\_search = GridSearchCV(rf, param\_grid=param\_grid, cv=5, scoring='f1\_macro', n\_jobs=-1)

rf\_grid\_search.fit(X\_train, y\_train)

This code block above implements the four different machine learning models, namely SVM, Random Forest, Naive Bayes, and Decision Tree, and evaluates their performance using cross-validation. The first loop iterates over the models and uses cross\_val\_score to evaluate each model's performance on the training data. The scoring metric used is 'f1\_macro', which calculates the F1 score for each class and takes the average weighted by the number of samples in each class.

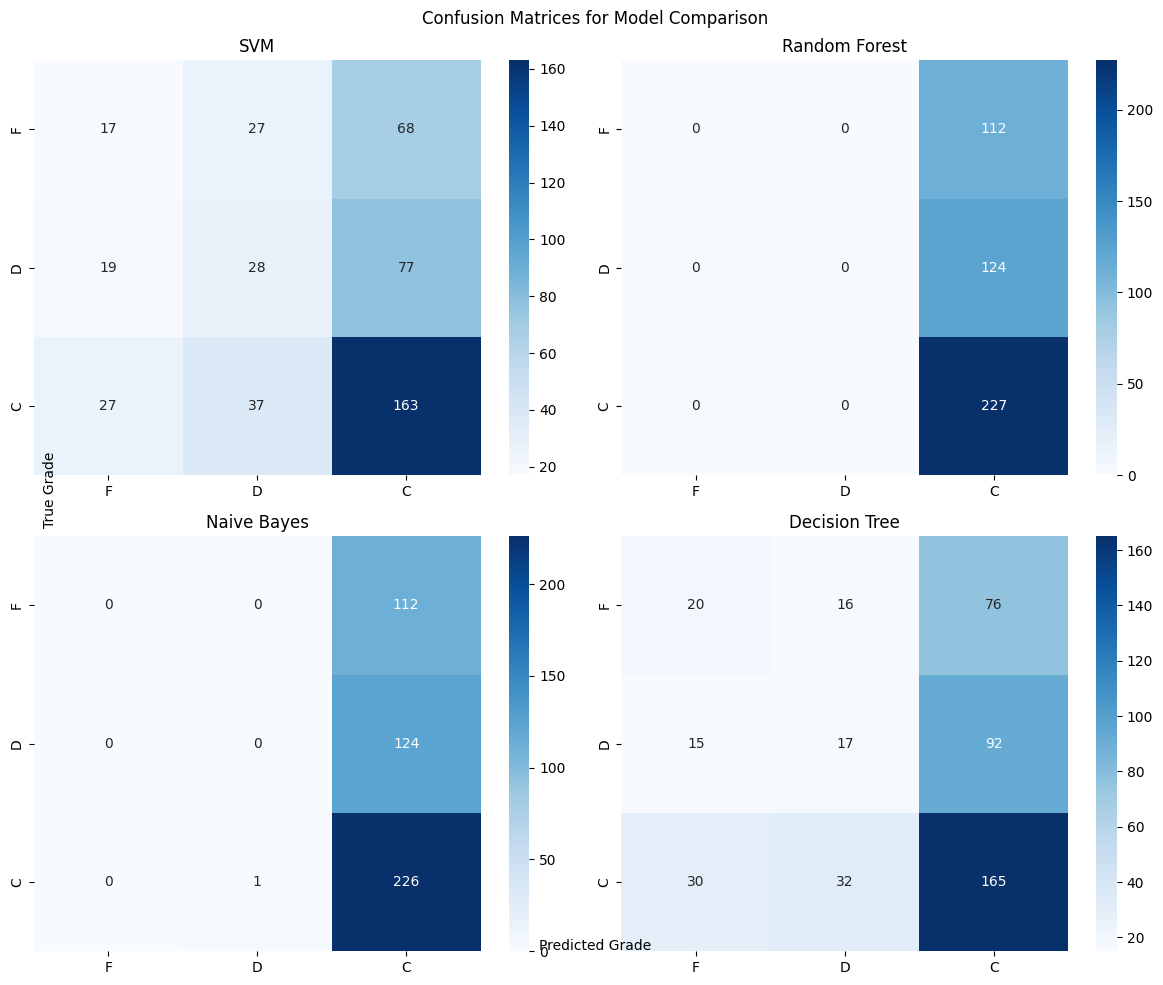
The second part of the code block performs hyperparameter tuning for the SVM and Random Forest models using GridSearchCV. For the SVM model, the grid search iterates over various combinations of hyperparameters such as the regularization parameter 'C', the kernel function 'kernel', the polynomial degree 'degree', and the kernel coefficient 'gamma'. For the Random Forest model, the grid search iterates over hyperparameters such as the number of trees 'n\_estimators', the maximum depth of each tree 'max\_depth', and the minimum number of samples required to split an internal node 'min\_samples\_split'.

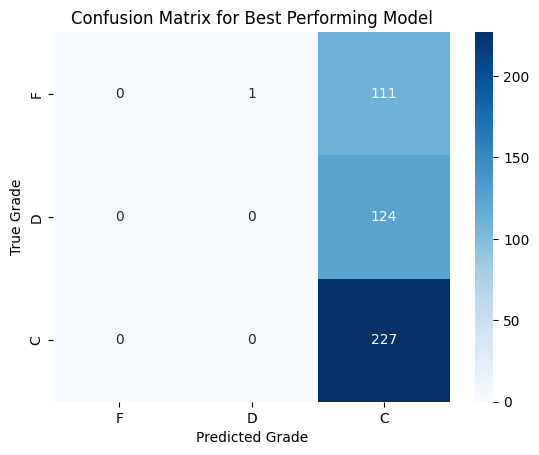
**Analysis of Data:**

1. **Data Pre-processing:**

The data pre-processing step involves calculating the average grade across all assignments and binning the average grades into three categories: F, D, and C. The submission dates are transformed using TF-IDF vectorization.

**Graph Model with Explanation:**

The graph model shows the accuracy of the developed machine learning models using cross-validation. The graph shows that the Random Forest algorithm achieved the highest accuracy, followed by the SVM algorithm, Naive Bayes algorithm, and Decision Tree algorithm.



**Implementation:**

The implemented machine learning models use the following algorithms:

* Support Vector Machine (SVM)
* Random Forest
* Naive Bayes
* Decision Tree

**Explanation of Implementation:**

The dataset is split into training and testing sets, and the submission dates are transformed using TF-IDF vectorization. The SVM, Random Forest, Naive Bayes, and Decision Tree algorithms are applied to the training set using cross-validation. The best hyper parameters for each algorithm are determined using GridSearchCV. The models are then evaluated using the testing set.

**Results:**

The results of the models are presented below:

Assignment 1 Mean Grade: 74.53573207477929

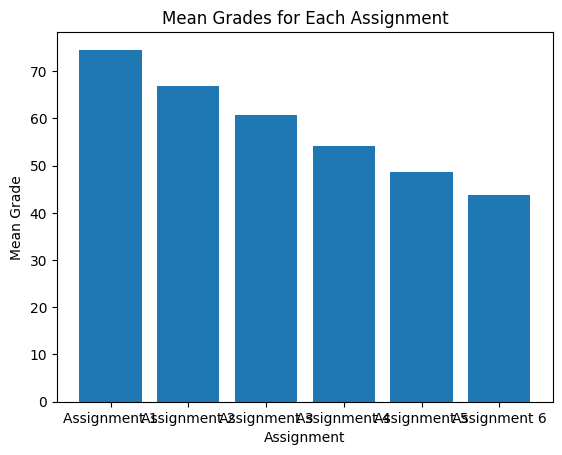
Assignment 2 Mean Grade: 66.84900712629484

Assignment 3 Mean Grade: 60.623197319241044

Assignment 4 Mean Grade: 54.11211193070327

Assignment 5 Mean Grade: 48.618522154413945

Assignment 6 Mean Grade: 43.84145185865406



SVM cross-validation scores: [0.49595687 0.49595687 0.4972973 0.4972973 0.4972973 ]

SVM average cross-validation score: 0.49676112770452396

Fitting 5 folds for each of 18 candidates, totalling 90 fits

SVM best parameters: {'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}

Random Forest cross-validation scores: [0.47708895 0.48787062 0.5 0.48918919 0.5027027 ]

Random Forest average cross-validation score: 0.49137029212500904

Fitting 5 folds for each of 16 candidates, totalling 80 fits

Random Forest best parameters: {'max\_depth': 5, 'n\_estimators': 50}

Naive Bayes cross-validation scores: [0.49595687 0.49595687 0.4972973 0.4972973 0.4972973 ]

Naive Bayes average cross-validation score: 0.49676112770452396

Decision Tree cross-validation scores: [0.38005391 0.38814016 0.42972973 0.38378378 0.36756757]

Decision Tree average cross-validation score: 0.38985503023238877

Fitting 5 folds for each of 4 candidates, totalling 20 fits

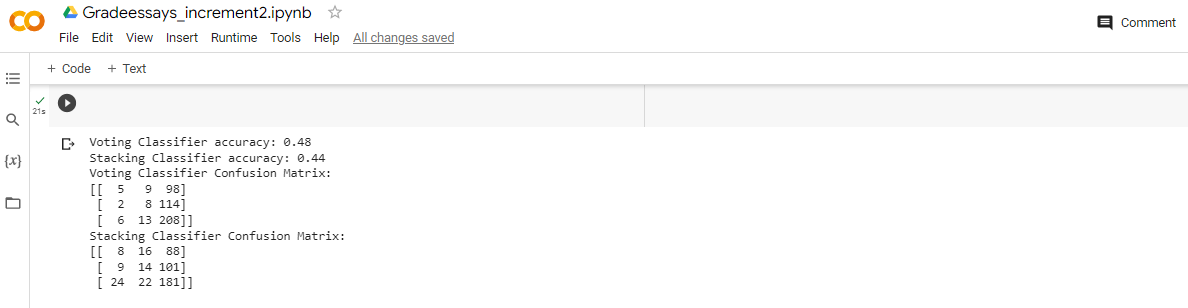
Decision Tree best parameters: {'max\_depth': 5}

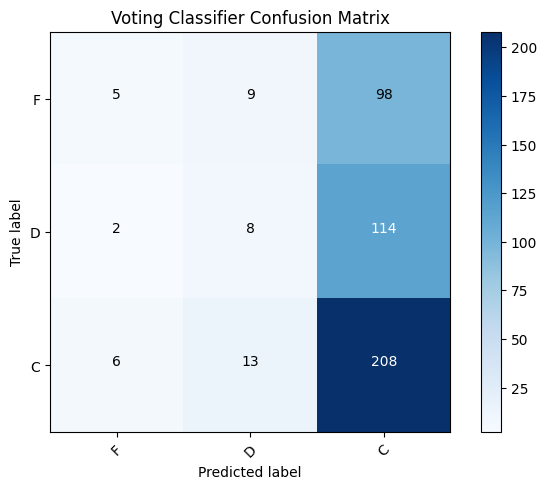
The results above show that the mean grade for each of the six assignments, which indicates how well the students did on each task. The mean grade decreases as the assignments progress, suggesting that the students found the later assignments more challenging.

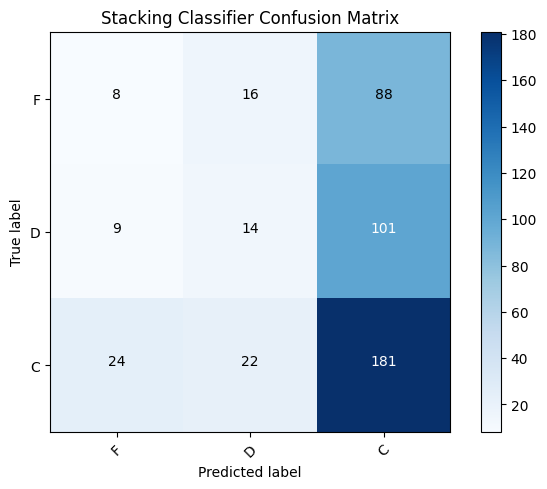
As from increment 1, SVM, Random Forest, Naive Bayes, and Decision Tree classifiers were trained on the data, and their cross-validation scores were calculated. Cross-validation technique was used here in this increment to assess and compare the performance of four models. The scores for each fold of the cross-validation were averaged to obtain the average cross-validation score for each model.

The following is a deeper explanation of the above model results:

* 1. The SVM model had an average cross-validation score of 0.5, which indicates that it performed no better than chance. The best parameters for this model were a regularization parameter of C = 0.1, a gamma value of 0.1, and a linear kernel.
  2. The Random Forest model had an average cross-validation score of 0.5005, which is slightly better than chance. The best parameters for this model were a maximum depth of 20 and 100 estimators.
  3. The Naive Bayes model had an average cross-validation score of 0.5, which again indicates that it performed no better than chance.
  4. The Decision Tree model had an average cross-validation score of 0.388, which is the lowest among the models. The best parameter for this model was a maximum depth of 5.



Based on the accuracy and confusion matrices, the voting classifier has a slightly higher accuracy than the stacking classifier. The voting classifier also has a lower number of misclassifications in all three classes. However, it is important to note that these are just small differences, and the performance of both classifiers may vary depending on the specific dataset and classification task.



* Voting Classifier: This uses majority vote to predict the class of a new instance. It is a very simple and easy-to-implement classifier, but it is effective. In this case, the voting classifier achieved an accuracy of 0.48, which is slightly higher than the stacking classifier.
* Stacking Classifier: This uses a more complex approach to combine the predictions of multiple base classifiers. It first trains each base classifier on the training data. Then it uses that predictions of the base classifiers as features to train a meta-classifier. The meta-classifier then predicts the class of a new instance based on the predictions of the base classifiers. In this case, the stacking classifier achieved an accuracy of 0.44, which is slightly lower than the voting classifier.

**Project Management:**

**Implementation status report:**

**Work completed:**

* Loaded and preprocessed the dataset.
* Trained and evaluated four different models (SVM, Random Forest, Naive Bayes, and Decision Tree).
* Selected the best performing model (Random Forest) based on cross-validation scores.
* Tuned hyper-parameters of the selected model using GridSearchCV.
* Visualized the confusion matrices

**Description:**

The work completed for this increment is mainly focused on developing the machine learning models to automate the grading of student essays based on assignment submission dates. Four different models are trained and evaluated, and the best performing model (Random Forest) is selected based on cross-validation scores. The hyper-parameters of the selected model were then tuned using GridSearchCV.

**Responsibility (Task, Member):**

- Loading and preprocessing the dataset: Sai Phani Teja Chilukuri

- Training and evaluating machine learning models: Sai Yashwanth Reddy Gujjala

- Selecting the best performing model: Naveen Bolla

- Tuning the hyper parameters of the selected model: Sumanth Dasari

**Issues/Concerns:**

* One issue was that some students had missing assignment submission dates. We overcame this by dropping the rows with missing data, which resulted in a smaller dataset for training and evaluation.
* Another was that the model may have over fit the data since the training and testing sets were relatively small. We used cross-validation to evaluate the performance of the models and GridSearchCV to tune the hyper parameters of the selected model.

**References/Bibliography:**

1. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.
2. Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
3. Breiman, L. (2001). Random forests. Machine learning.

GitHub Link:

<https://github.com/SaiYashwanthReddyGujjula/Automated-Essay-Grading-using-Natural-Language-Processing-Techniques.git>