PROBLEM STATEMENTS AND APPROACH

Digital technologies are becoming more common in education, allowing students all around the world to access individualized, high-quality education resources. Importantly, I have been some given diagnostic questions among these resources: the responses students give to specific questions offer key information about the exact kind of misconceptions that the students may hold. Analyzing huge amounts of data generated by students' interactions with these diagnostic questions will help educational management to better understand the students' learning state and, as a result, allow them to automate learning curriculum recommendations. In this project, I focus on students' information and answer records to these multiple-choice diagnostic questions in order to:

1) properly predicts student responses - Right or Wrong (to predict whether a student answers a question correctly.). This task is a binary-class prediction problem.
2) accurately predicting student response – answer prediction (to predict which answer a student gave to a particular question). The questions in the dataset are all multiple-choice, each with 4 potential choices and 1 correct choice, so this task is a multi-class prediction problem.

Data: I make use of two major Data which are Primary Data and Metadata

Primary Data.

This is main training data, consisting of records of answers given to questions by students. It can be found in the files train task 1 2.csv and train task 3 4.csv The columns are as follows:

• QuestionId: ID of the question answered.

• UserId: ID of the student who answered the question.

• AnswerId: Unique identifier for the (QuestionId, UserId) pair, used to join with associated answer metadata

• IsCorrect: Binary indicator for whether the student's answer was correct (1 is correct, 0 is incorrect).

• CorrectAnswer: The correct answer to the multiple-choice question (value in [1,2,3,4]).

• AnswerValue: The student's answer to the multiple-choice question (value in [1,2,3,4]).

Metadata

This contain different data such as Question Metadata, Student Metadata, Answer Metadata

Question Metadata contains:

• SubjectId

•Question content

Student Metadata contains:

• UserId: An ID uniquely identifying the student, which can be joined to the primary dataset.

• Gender: The student's gender, when available. 0 is unspecified, 1 is female, 2 is male and 3 is other.

• DateOfBirth: The student's date of birth, rounded to the first of the month.

• PremiumPupil: Whether the student is eligible for free school meals or pupil premium due to being financially disadvantaged.

Answer Metadata contains:

The following metadata is provided about each individual answer record in the dataset:

• UserId: An ID uniquely identifying the answer, which can be joined to the primary dataset.

• DateAnswered: Time and date that the question was answered, to the nearest minute.

• Confidence: Percentage confidence score given for the answer. 0 means a random guess, 100 means total confidence.

• GroupId: The class (group of students) in which the student was assigned the question.

• QuizId: The assigned quiz which contains the question the student answered.

• SchemeOfWorkId: The scheme of work in which the student was assigned the question.

Solving this problem, I merged both primary data and metadata through some unique identifier so as to generate more features for effective training. I train my datasets on different algorithm for each task as seen below, this is because the nature of the problem is different. Task 1 is a binary classification while Task 2 is a multi-class classification.

TASK 1:

o SVM- 3 different predications using different kernels and hyper parameters E.g. Linear, rbf and poly I also include rbf kernel with feature selections

o Naive Bayes classifier (Gaussian and Bernelious)

o Logistic regression

o Decision tree (3 different trees with different hyperparameters)

o Random Forest (use Grid search to find best parameters:

✓ For the criterion (Gini or entropy)

✓ Max feature

✓ Max depth

✓ Number of trees

o Gradient Boosting Classifier (GridSearchCV for Hyperparameter Tuning)

o Bagging Classifier (one with Naïve bayes and one with decision trees)

o AdaBoosting (GridSearchCV for Hyperparameter Tuning)

o Then I add my own method for predication (XGBClassifier) that i believe it outperform the above predications

o Then I Use Ensembling method to improve the results

➢ I then Compare the above classifiers methods according to Accuracy, Precision, Recall, F1-Score, and ROC.

➢then I Ploted the comparisons!

The major evaluation metric I used is prediction accuracy where I was able to use other metrics such as ROC_AUC, F-1 Score, Recall, Precision for my accuracy evaluation. The accuracy is the number of predictions that match the true correctness indicator, divided by the total number of predictions (in test set)

Accuracy = correct predictions/ total predictions

Though in the case of task 2 I still use the same metric prediction accuracy as above, except that the true answers are now categorical instead of binary.

```
# TASK 1 Best Model
```

Adaboosting Classifier gave me the best model having an accuracy of 95%.
This was done using feature selections. The Feature selection techniwue used was the Pearson correlation which helps to get most correlated features with the target variable called IsCorrect.

After selecting most correlated features, then i used decision tree estimators to get best model with hyperparameters tuning that help to optimize my model training and learning.
those hyper parameters used are;
Criterion, splitter, n_estimators, max_features, class_weight, max_depth

I then performed grid search on this hyperparameter to obtain optimal results. the parameter passes are;

```
parameter= {"base_estimator__criterion": ["gini", "entropy"],
"base_estimator__splitter": ["best", "random"],
"n_estimators": [1, 2,3,5]}
```
After performing this grid search, my accuracy result was very low (83%) with max_depth of 3 So I kept tuning the max_depth to a value of 15 which then gave me accuracy of 95%

# ADABOOSTING ARCHITECTURE

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method . The most common algorithm used with AdaBoost is decision trees with one level that means with Decis ion trees with only 1 split. These trees are also called Decision Stumps.

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then a ssigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lowe error is received.

STEP1. First of all, our data points will be assigned some weights. Initially, all the weights will be equal.

SEP 2; We start by seeing how well each feature classifies the samples.
Then we'll create a decision stump for each of the features and then calculate the Gini Index of each tre e. The tree with the lowest Gini Index will be our first stump.

Step 3 Then it now calculates the "Amount of Say" or "Importance" or "Influence" for this classifier in cla ssifying the datapoints using some formula:
The total error is nothing, but the summation of all the sample weights of misclassified data points.

Step 4 – Once the Total Error has been calculated as well as the and performance of a stump. Then it will start update the weights because if the same weights are applied to the next model, then the output re ceived will be the same as what was received in the first model.

The wrong predictions will be given more weight whereas the correct predictions weights will be decrea sed. So the next model will be build after updating the weights, more preference will be given to the poi nts with higher weights.
Then it finds the importance of the classifier and total error needed to finally update the weights.

Step 5 – It then make a new dataset to see if the errors decreased or not. In doing this it will remove the "sample weights" and "new sample weights" column and then based on the "new sample weights" it wil l divide the data points into buckets.

Step 6 – the algorithm does selects random numbers from 0-1. Since incorrectly classified records have higher sample weights, the probability to select those records is always very high.
Suppose the 5 random numbers our algorithm take is 0.38,0.26,0.98,0.40,0.55.
These random numbers fall in the bucket and according to it, and will make a new dataset. This comes out to be our new dataset.

Step 7 – Now this act as our new dataset and we need to repeat all the above steps i.e.

1.Assign equal weights to all the data points
2.Find the stump that does the best job classifying the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index
3.Calculate the "Amount of Say" and "Total error" to update the previous sample weights.
4.Normalize the new sample weights.
The algorithm Iterate through these steps until and unless a low training error is achieved.

Suppose with respect to our dataset we have constructed 3 decision trees (DT1, DT2, DT3) in a sequentia l manner. If we send our test data now it will pass through all the decision trees and finally, we will see which class has the majority, and based on that we will do predictions for our test dataset.

# TASK 2 Best Model
Two Model gave the best Model: which are
a. ENSEMBLE METHOD
b. KNN CLASSIFIER

Ensembling Techniques gave the best Results:
Three differents classifier was passed as an estimator into the ensemble method. The three estimator us ed are;
Before I achived these 3 best estimators, I have used different numbers of estimators such as logisticReg ression, svm, Adaboost, gradient Boosting but yest they were notgiving an optimum accuracy suitable for me.

After so much tunning and i was able to get the 3 best estimators that gave the best accuracy. These 3 estimators are;
1. KNN with a n_neighbors parameter of 3
2. Decision tree with a criterion parameter of entropy
3. Naive Bayes

After thatm i performed cross validation on the 3 estimators thereby helping my model to behave very. This cross validation behaves well with CV of 2

The type of ensemble technique used was VotingClassifier. These estimator was then passed into Voting Classifier.

Then the VotingClassifier helped to get an accurcy of 88% which was far better than every other classifier asides KNN and far better than KNN because it takes a lesser time to learn as compared to KNN

2. KNN
KNN gave a good result f 88% accuracy_score
KNN has n_neighbors parameters that can be tuned to get the best results. I initially started with n_neighbiors of 5 which gave a very good performance of 83% accuracy_score
Then i have to plot the error rate so as to minimize the error rate using different values of n_neighbors parameter ranges from 1-40.
During this process of iteration and learning at different n_neighbors. The error was very mininimized at n_neighbors of k = 36.
Then using K of 36 gave us the best accuracy of 88%

# Architecture OF KNN MODEL

KNearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning t echnique.
KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
KNN algorithm stores all the available data and classifies a new data pointbased on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
KNN algorithm can be used for Regression as well as for Classification butmostly it is used for the Classifi cation problems. Reasons it is most suitbale for out AnswerValue Task
KNN is a nonparametric algorithm, which means it does not make any assumption on underlying data.
It is also called a lazy learner algorithm because it does not learn from the training set immediately inste ad it stores the dataset and at the timeof classification, it performs an action on the dataset.
KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
Example: in our task 2, we have an AnswerValue similar t a multi class of 4 classes 1,2,3,4, and we want to know either the AnswerValue Selected by the student is 1,2,3,4. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to either the 1,2,3,4 values and based on the most similar features it will put it in either 1,2,3,4 categories.

HOW KNN WORKS.

The K-NN working can be explained on the basis of the below algorithm:
Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of K number of neighbors
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number

Of the neighbor is maximum.
Step-6: Our model is ready

e.g.
Suppose we have a new data point from a students and we need to put it in the required category.
Firstly, we will choose the number of neighbors, which i chose the k=5.
Next, I calculated the Euclidean distance between the data points. The Euclidean distance is the distance between two points.
By calculating the Euclidean distance, we got the nearest neighbors, i.e
certain nearest neighbors in category 1, certain nearest neighbors in category 2, certain nearest neighbors in category 3 and certain nearest neighbors in category 4. so the new data poi
nt must belong to either of the 4 categories.

SELECTING BEST VALUYE OF K
There is no particular way to determine the best value for "K", so we needto try some values to find the best out of them. The most preferred value for K is 5 and that was what gave me 83% at first. before trying different value of K.
A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
Large values for K are good, but it may find some difficulties. But in my model K value of 36 gave me the best accuracy which in turned improve my, model performances.

Advantages of KNN Algorithm:
It is simple to implement.
It is robust to the noisy training data
It can be more effective if the training data is large.
Disadvantages of KNN Algorithm:
Always needs to determine the value of K which may be complex some time.
The computation cost is high because of calculating the distance between the data points for all the training samples.