AI POWERED EDUCATION SYSTEM

Using Python

By Udit Pragadeesh

CONTENTS

INTRODUCTION

Every student is different and analyzing each student individually is a task on its own. Where human effort is not enough or is highly required, we can use AI models to reduce pressure on humans. This project is a step towards helping teachers in analyzing and understanding students. It helps understanding students easier and students learn better.

This app has different features to analyze each student's performance, learning style, skills etc. to better guide them. These are made using trained Deep learning and Machine Learning models.

Use Cases:

- EdTech companies can integrate the solution into their LMS to provide customized learning paths.
- Schools and colleges can use the assistant for remedial coaching and curriculum support.
- Parents gain insights into their child’s learning behavior and can take early action.

TOOLS & LIBRARIES USED

- **Python** – programming language
- **Visual studio code** – IDE
- **Pandas, NumPy** – Data Handling
- **Seaborn, Matplotlib** – Data Visualization
- **Scikit Learn** – Machine learning Framework
- **PyTorch** – Deep learning Framework
- **Transformers/Hugging face** – Pretrained LLM models
- **Streamlit** – App Execution
- **Kaggle** – Data source
- **SQLite/sqlite3** – Query and database
- **Pickle**- Saving models

Python

Python is a high-level, general-purpose programming language. It is used as the programming language for complete project, from data handling, cleaning, visualization, preprocessing to execution.

Why python? Python is one of the most popular programming languages today. It has many advantages which makes it an excellent choice for data science.

1. **Easy to learn and use**
   Python does not have complex syntax or codes. It has readable close to common English language codes and a simple easy to use and remember syntax. Which makes it suitable for beginners to learn.
2. **Huge libraries**
   Python features a large number of libraries that can be easily downloaded and accessed. Libraries help save functions and classes for use in different project which makes it efficient.
3. **Large community**
   Large community means many tutorials. There are people around the world available for help and support. It also adds up to new innovations and libraries adding to inventory.
4. **Versatile and Powerful**

   It is used in various domains like

   - Data Science/ Machine Learning
   - Web Development

- Game Development etc.

Visual Studio Code

Visual Studio code commonly referred to as VS Code is an integrated development environment developed by Microsoft. It is available in Windows, Linux, macOS and Web browsers which makes it cross-platform and available to use. It supports multiple languages like **Python, C#, C++, JavaScript, HTML** and so on this makes it versatile.

VS Codes comes with direct access to terminals which makes it easy to access python and computer. It also has many debugging and support features which makes coding efficient and easy.

Pandas, NumPy

Most data used for this project are tables and csv files. Pandas is efficient and common way of accessing and storing tables in python. Pandas comes with a **'read_csv'** function to directly load csv files as a **DataFrame** in python. Which can be later used for visualizations in EDA, cleaning, and manipulation.

NumPy is a python Library used majorly for working with arrays and performing mathematical and statistical operations. In this project it is used for data manipulation and data transformations.

Seaborn, Matplotlib

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for making attractive and informative statistical graphs and charts. Almost all visualizations done in this project are through seaborn.

Matplotlib is an open-source visualization library for Python programming language, widely used for creating static, animated and interactive plots. Matplotlib especially **pyplot** is used in this project for editing plots made using seaborn.

Scikit Learn

Scikit-learn also known as sklearn is widely-used open-source Python library for machine learning. It is built on other scientific libraries like NumPy, SciPy and Matplotlib to provide efficient tools for predictive data analysis and data mining. It offers a consistent and simple interface for a range of supervised and unsupervised learning algorithms, including classification, regression, clustering, dimensionality reduction, model selection and preprocessing.

Commonly used algorithms from sklearn are **Linear regression, Logistic Regression, Decision Trees, Support vector machine, Naive bayes, K nearest neighbor, Random Forest,** and **Gradient boost** for supervised learning. **K-means, DB Scan and Hierarchical** clustering algorithms for unsupervised learning. It also features various preprocessing and model selection algorithms like **Standard scaler and train test split** etc.

PyTorch

PyTorch is an open-source deep learning framework designed to simplify the process of building neural networks and machine learning models. With its dynamic computation graph, PyTorch allows developers to modify the network's behavior in real-time, making it an excellent choice for both beginners and researchers.

Transformers/Hugging face

Hugging Face is a community where people collaborate to create and improve ML models. It provides the tools and resources for deploying, training and using ML models, making complex technologies accessible to everyone. It is best known for its Transformers library, which simplifies ML tasks like natural language processing (NLP) by providing pre-trained models. It is often referred to as the GitHub of machine learning. It is a place where users can share, discover and build with the help of the community, making it easier to work with AI tools. Hugging Face is a leading player in machine learning, offering open-source tools and fostering collaboration in the AI community.

Streamlit

The trend of Data Science and Analytics is increasing day by day. From the data science pipeline, one of the most important steps is model deployment. We have a lot of options in python for deploying our model. Some popular frameworks are Flask and Django. But the issue with using these frameworks is that we should have some knowledge of HTML, CSS, and JavaScript. Keeping these prerequisites in mind, **Adrien Treuille, Thiago Teixeira, and Amanda Kelly** created "Streamlit". Now using streamlit you can deploy any machine learning model and any python project with ease and without worrying about the frontend. Streamlit is very user-friendly.

Kaggle

Kaggle is a data science competition platform and online community for data scientists and machine learning practitioners under Google LLC. Kaggle enables users to find and publish datasets, explore and build models in a web-based data science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

SQLite

SQLite is a lightweight, self-contained database engine that is easy to use and does not require a separate server to operate. It is embedded into applications, making it ideal for mobile apps, small websites, and projects where simplicity and speed are crucial.

Unlike other databases, SQLite stores all data in a single file, making it portable and easy to manage. Despite its small size, it supports most SQL standards, allowing developers to perform complex queries and transactions efficiently. SQLite is reliable, fast, and perfect for many everyday database needs without the overhead of a full-fledged database system.

I have used sqlite3 library in python to access and query SQLite database through python.

Pickle

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What Pickle does is it **“serializes”** the object first before writing it to a file.

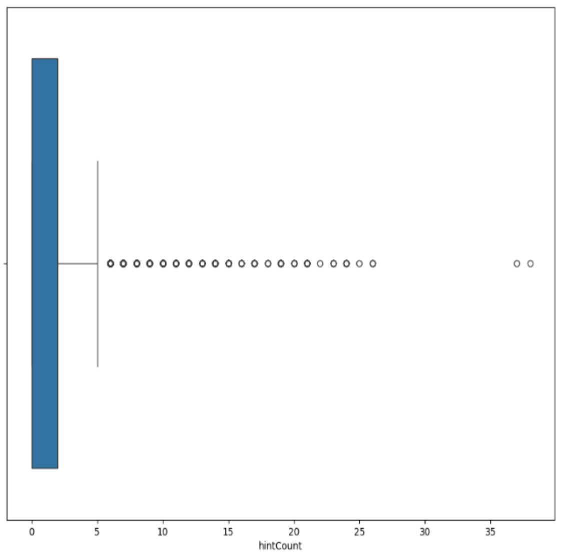
Pickling is a way to convert a Python object (list, dictionary, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another Python script. It provides a facility to convert any Python object to a byte stream. This Byte stream contains all essential information about the object so that it can be reconstructed, or "unpickled" and get back into its original form in any Python.

PASS/FAIL PREDICTOR

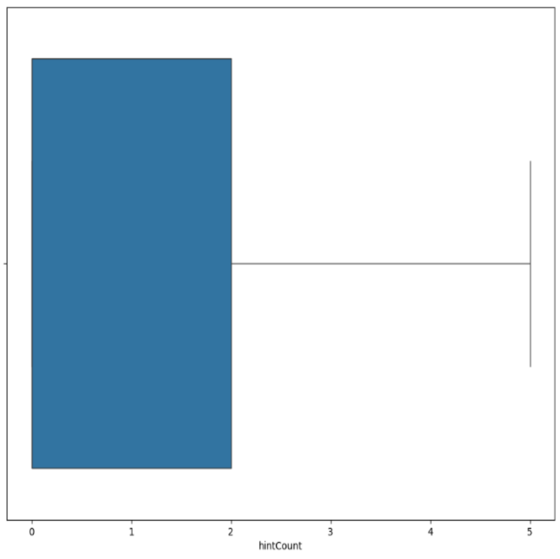
EDA & FEATURE SELECTION

Multiple csv files of student performances on different tests. It includes student skills, problem type, correct, hint count, attempt count and so on. On loading and combining all files total number of data adds up to 9,24,816 rows and 76 columns. Considering problem statement and ease of analysis 9 columns were selected and used for further analysis.

- Seaborn boxplot was used to check for outliers in numerical columns. Number of outliers identified were low compared to total number of rows available, so they were removed. One of the plots before and after is shown below.

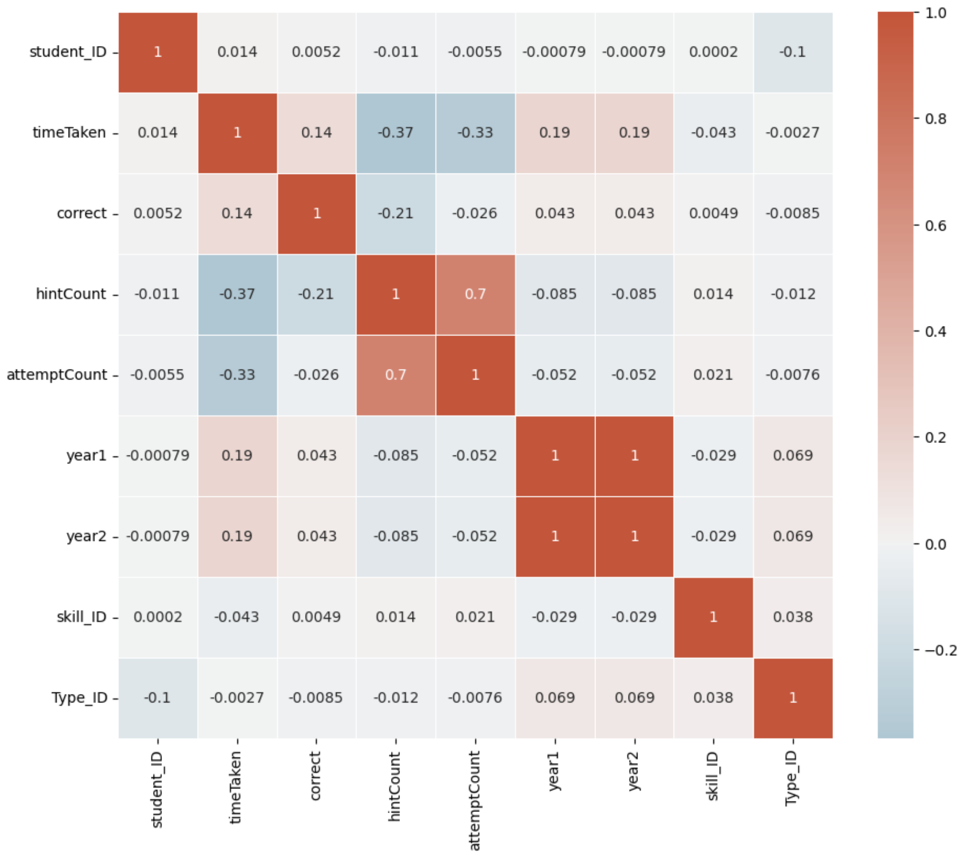


Hint count before removing outliers



Hint count after removing outliers

- Count plot was used to check if the distribution of data between problem types.
- Categorical data were label encoded using scikit learns label encoder. Then correlation between columns was analysed using heatmap.
- The data is then loaded into a data base for further use along with encoding data for future reuse.



Correlation heatmap for all columns selected

MODEL TRAINING AND EVALUATION

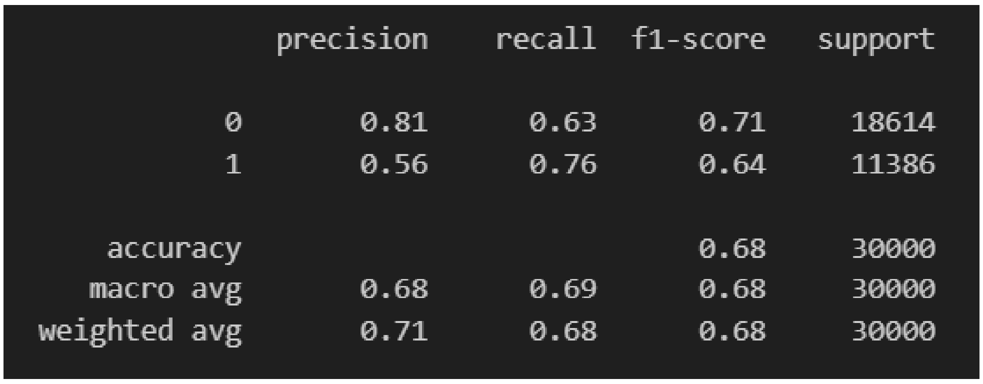
I chose to predict the number of correct answers form the number of questions and based on the percentage marks obtained compared to passing percentage a model can predict if a student can pass or fail. Since the model predicts whether between two classes it is a **binary classification** model.

For this binary classification model it was said in the problem statement to use supervised machine learning using scikit learn. I have considered the following models

- Linear regression
- Random forest classifier
- Support vector machine
    - LinearSVC
    - SVC

For evaluation of these models **Classification report** from sklearn classification metrics was used. The classification report provides a table of **recall, precision** and **f1 score** along with support. This provides a detailed evaluation scores for classification models.

After running training and evaluation of these different models and fine tuning **RandomForestClassifier** with default parameters is used as it provides best scores and accuracy compared to other models. This model was saved as "correct_predictor.sav" for further use using pickle library.



```
              precision    recall  f1-score   support

           0       0.81      0.63      0.71     18614
           1       0.56      0.76      0.64     11386

    accuracy                           0.68     30000
   macro avg       0.68      0.69      0.68     30000
weighted avg       0.71      0.68      0.68     30000
```

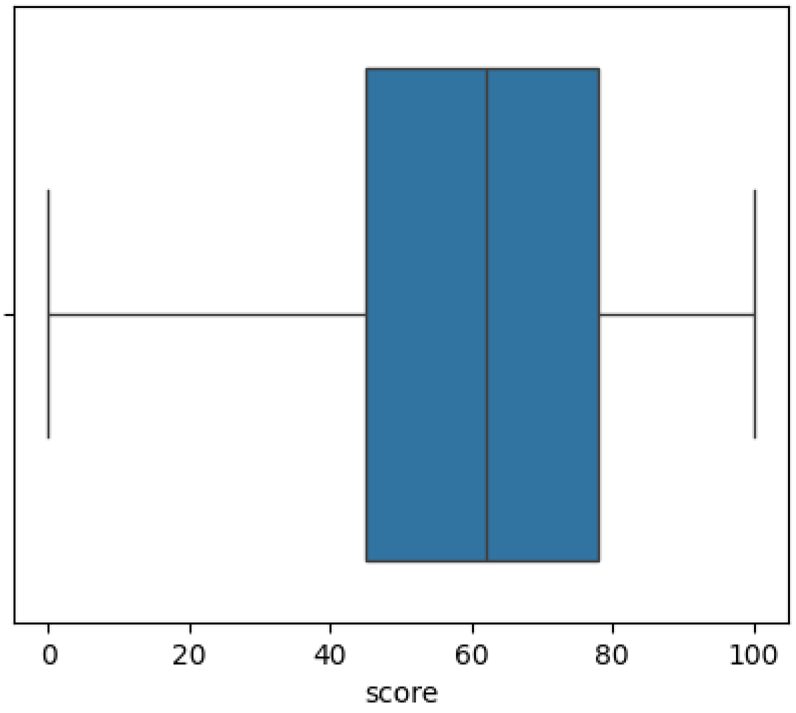
Classification report

SCORE PREDICTOR

EDA & FEATURE SELECTION

Data used for this model with 14000 rows of student IDs, skills, test IDs, problem types and performance of students in these tests. The data is clean and does not need to be cleaned and columns available are all required based on target.

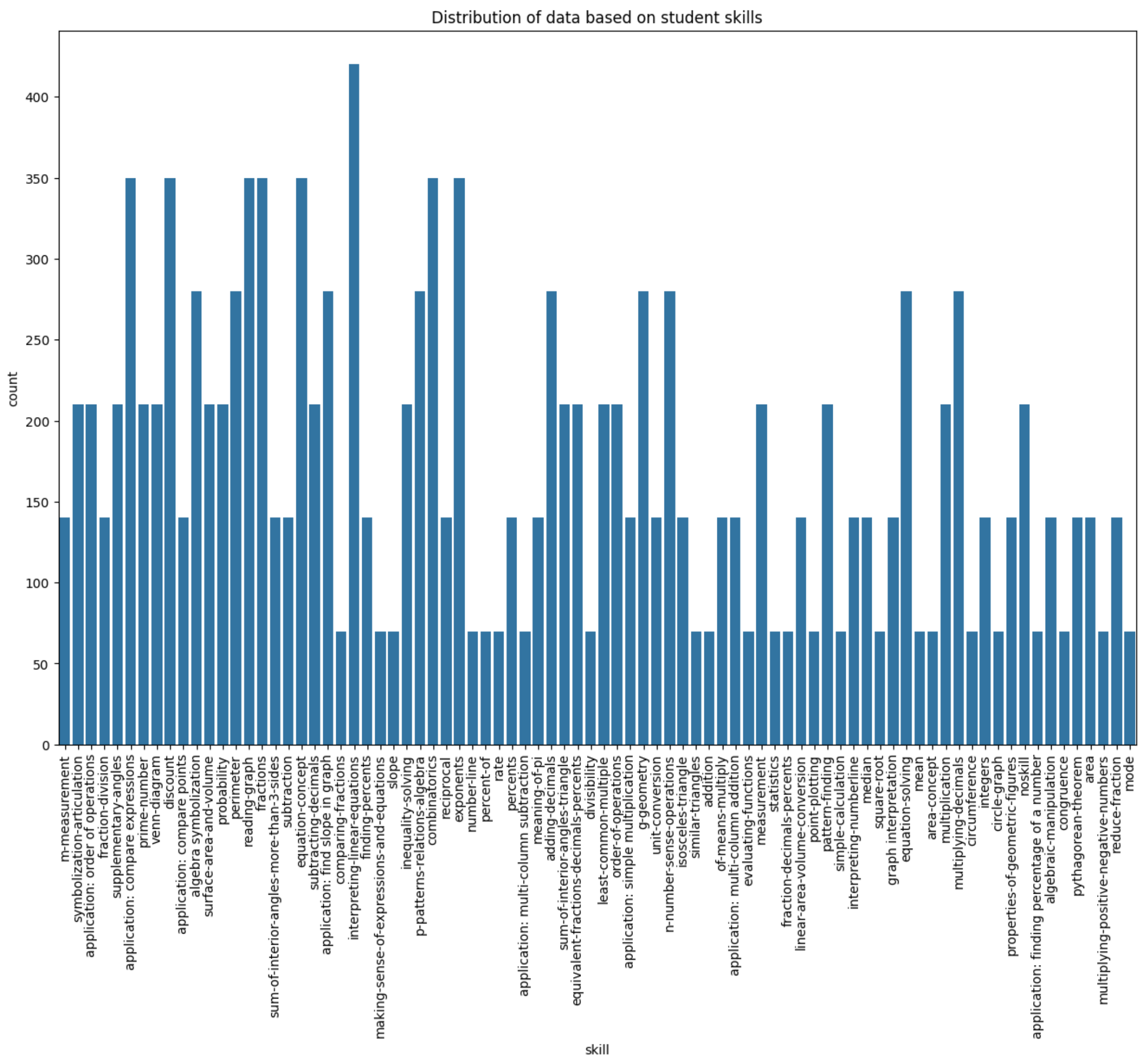
- First boxplots to check for outliers in numerical columns and no outlier was found in any of the numerical columns.



Boxplot of score column

- The data is checked if there is linear regression between Time taken column and score individually separating all students using seaborn lmplot but the regression was not linear.

- Distribution of data in categories was visualized using count plot.



- Student ID column was not used for this project as it does not impact the score.

MODEL TRAINING AND EVALUATION

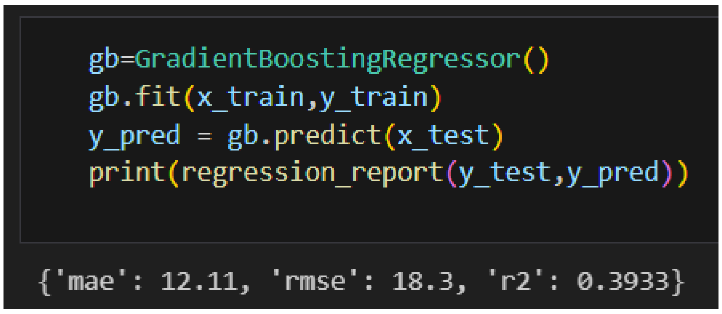
Categorical columns skills and Type are one hot encoded as these columns are not ordinal and one hot encoding provides better accuracy for using these features. Since this is a regression problem the following regressor models were used

- Decision Tree regressor
- Random forest regressor
- Gradient boosting regressor
- Xtreme gradient boosting regressor
- Adaboost regressor
- K nearest neighbor regressor
- Support vector regressor

The model was evaluated using the following metrics,

- Mean absolute error
- Root mean squared error
- R2 score

After training and testing our data on all above-mentioned models, even though it had slightly higher mean absolute error **GradientBoostingRegressor** with default parameters was selected since it had the lowest root mean squared error and the highest r2 score.



```
gb=GradientBoostingRegressor()
gb.fit(x_train,y_train)
y_pred = gb.predict(x_test)
print(regression_report(y_test,y_pred))
```

```
{'mae': 12.11, 'rmse': 18.3, 'r2': 0.3933}
```

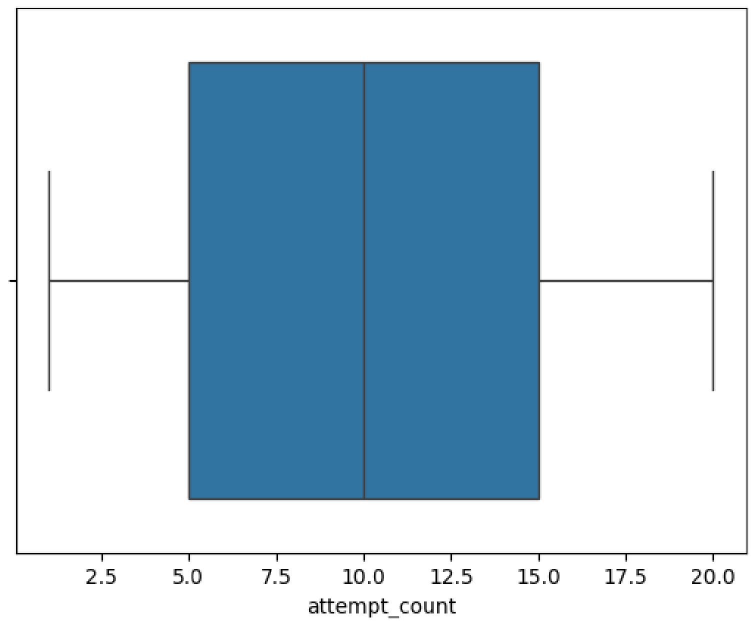
Evaluation result of selected model

Selected model and encoder are formed as a pipeline using sklearn pipeline and saved as ‘score_predictor.sav’ using pickle.

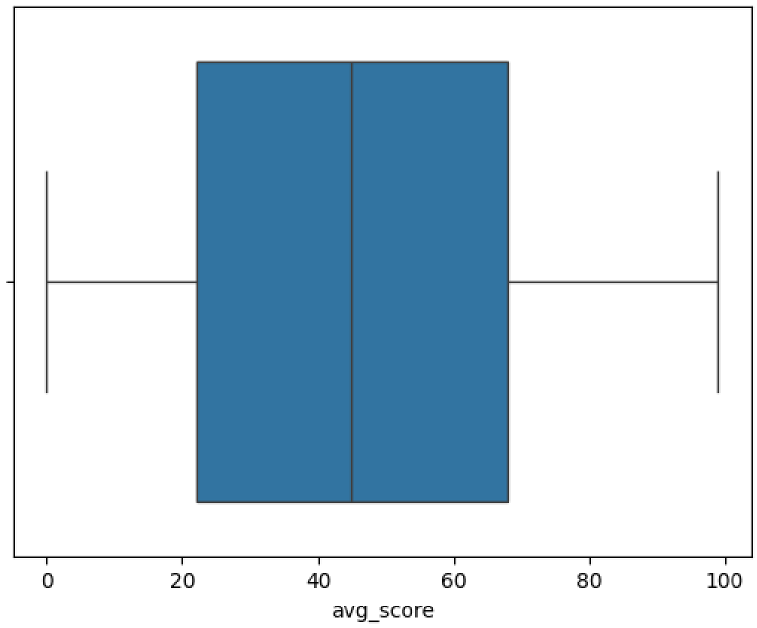
EDA & FEATURE SELECTION

This data has a detail on several different students with columns Student ID, attempt count, number of videos, number of books, time on topic, average score and average time taken.

- First box plot to check for outliers in numerical columns. Based on the plot no major outliers was found.

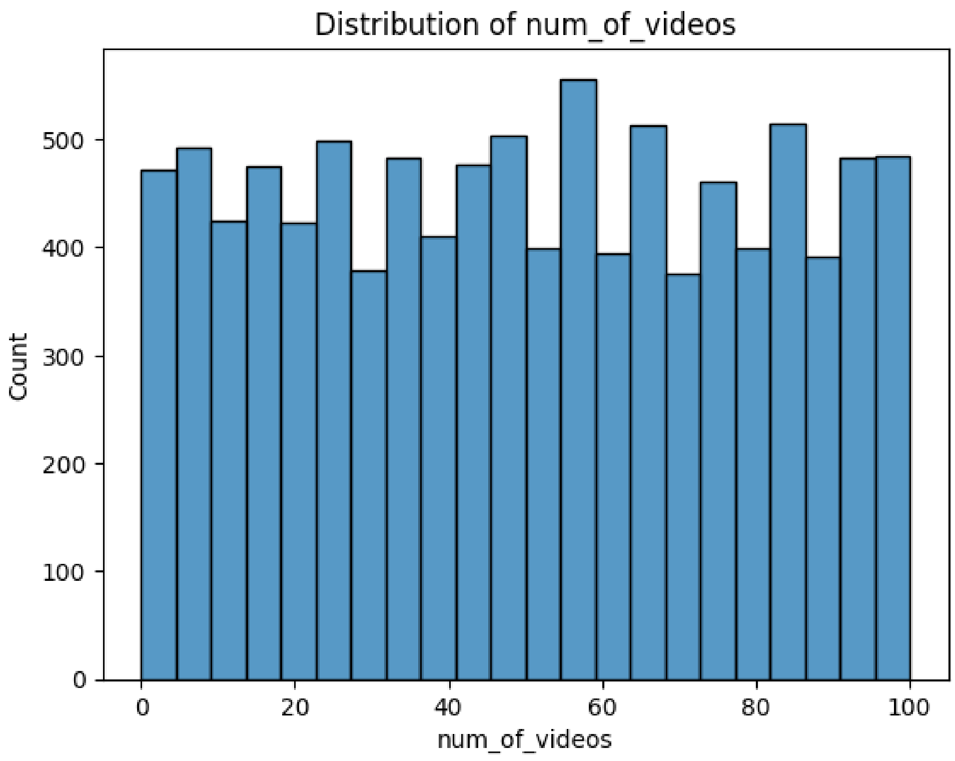


Boxplot of attempt count

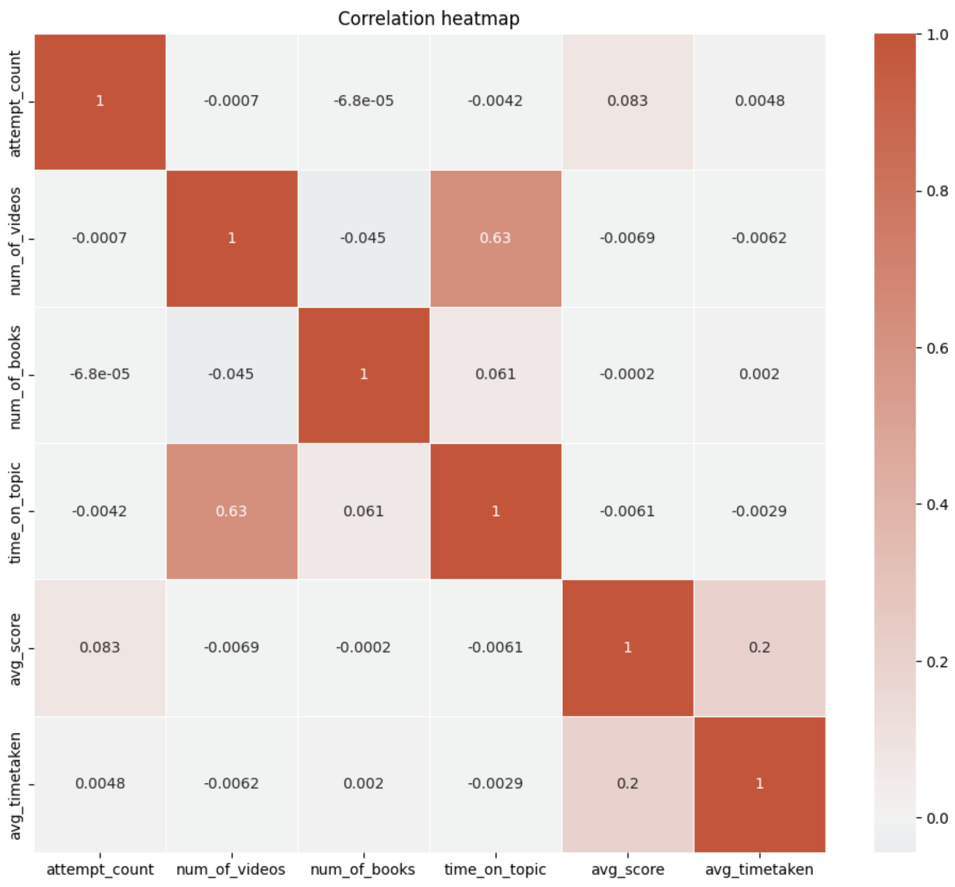


Boxplot of average score

- Histogram for all columns to check for the distribution of data is created using seaborn histplot.



- Correlation between columns was checked using heatmap.



	attempt_count	num_of_videos	num_of_books	time_on_topic	avg_score	avg_timetaken
attempt_count	1	-0.0007	-6.8e-05	-0.0042	0.083	0.0048
num_of_videos	-0.0007	1	-0.045	0.63	-0.0069	-0.0062
num_of_books	-6.8e-05	-0.045	1	0.061	-0.0002	0.002
time_on_topic	-0.0042	0.63	0.061	1	-0.0061	-0.0029
avg_score	0.083	-0.0069	-0.0002	-0.0061	1	0.2
avg_timetaken	0.0048	-0.0062	0.002	-0.0029	0.2	1

MODEL TRAINING AND EVALUATION

This problem does not have a target column and data needs to be separated into classes. Clustering a method of unsupervised machine learning is used. Sklearn provides 3 different clustering algorithms **KMeans, DB scan** and **Hierarchical** clustering.

The model was first scaled to get uniform variance across all columns so that cluster are not limited to one specific column. Then model was trained and tested using different number of clusters.

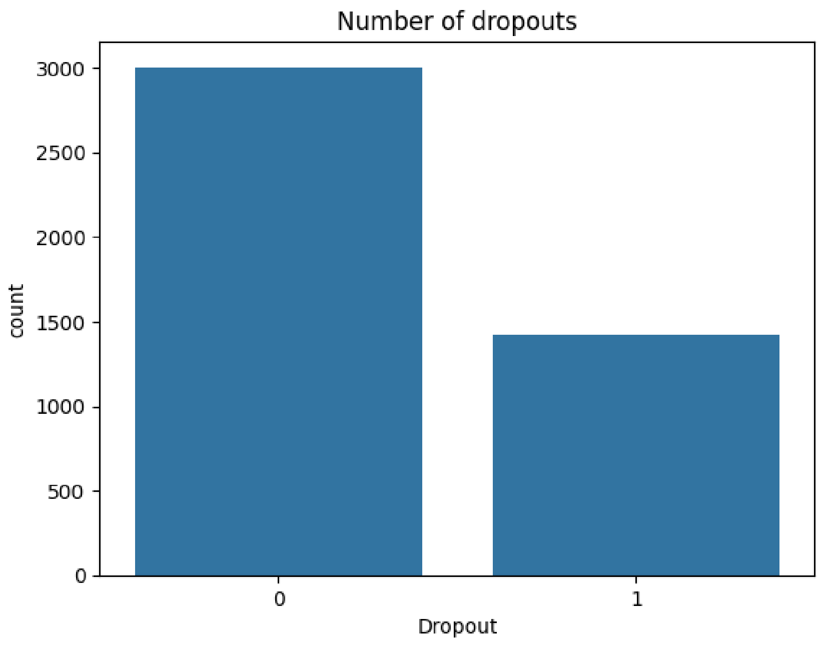
Evaluation of the models was done using silhouette score and considering the number of meaningful classes that can be formed. Based on testing KMeans with 32 clusters gave the best score and meaningful classes that can be formed from the data available.

Classes are then named and saved as csv file called 'cluster_names' for later use. The model is saved as 'Learning_style.sav' using pickle using pickle.

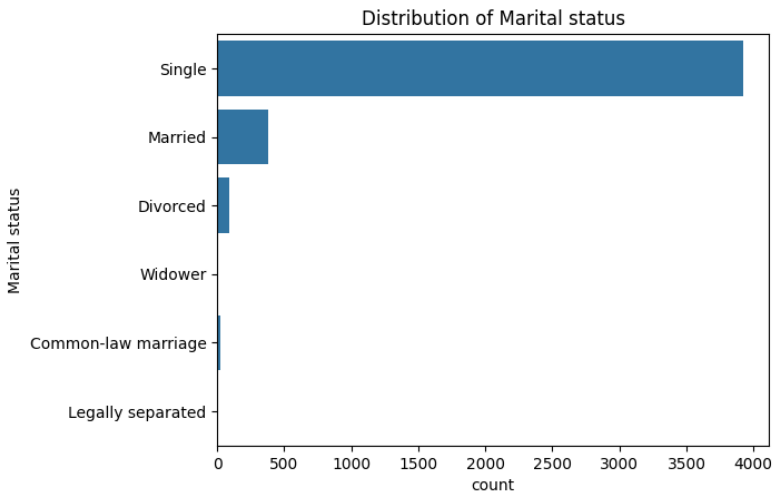
EDA & FEATURE SELECTION

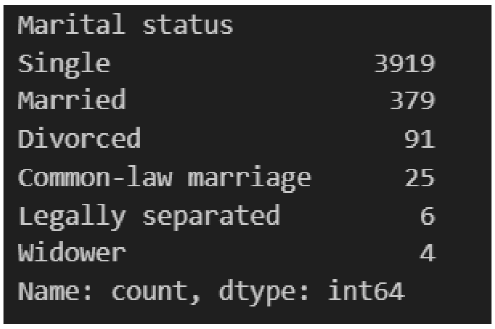
Data is tabular with 35 columns and 4,424 rows of data o students. The target column contains 3 different categories ‘Dropout’, ‘Graduate’ and ‘Enrolled’. But since problem statement requires risk of only drop out, New column ‘Dropout’ is created to show binary data.

Distribution of data between number of positives and negatives in dropout column was checked using count plot.



Then distribution of data between categories in all other categorical columns was checked by count plot and value counts.





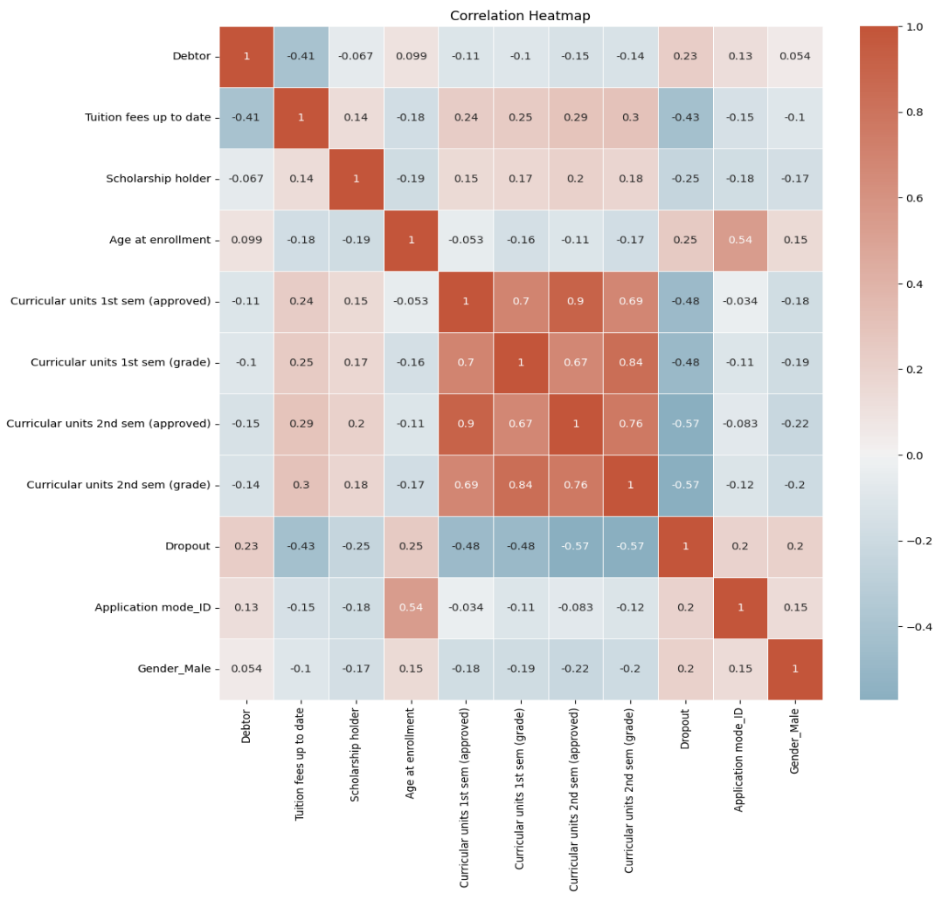
```
Marital status
Single                 3919
Married                 379
Divorced                 91
Common-law marriage      25
Legally separated         6
Widower                   4
Name: count, dtype: int64
```

Counts of Marital status

In these categorical columns if the count is less than 100 then these categories are all changed summed as others. Which makes the data more even with lesser number of categories. Categorical columns with multiple columns are label encoded and binary columns are all one hot encoded.

Numerical columns are checked for outliers using boxplot. Number of outliers in all these columns was also checked.

Columns with less significant correlation with dropout column are all removed from the dataframe and the modified dataframe was used for model training. Using heatmap correlation between column in this modified dataframe are visualized.



MODEL TRAINING AND EVALUATION

Deep learning model with a simple linear neural network with two hidden layers, **ReLU** activation function between these layers and **sigmoid** activation function at end for binary classification model. A drop out of 30% is given to regularize neurons to prevent overfitting. The Neural Network is created using **PyTorch** neural network.

Loss function used for this model is **BCE loss** (Binary Cross-Entropy loss) and optimizer is **SGD** (Stochastic Gradient Descant) with a learning rate of 0.001 and a weight decay of 0.0005 is used.

Data is prepared and ready to be loaded to the model using Dataset and Dataloader from torch library. Features are then scaled using standard scaler.

With model in train mode and optimizer set to zero gradient model is trained with 750 epochs to get good accuracy and lesser loss. Average loss in the BCE loss function is calculated and taken as training loss.Model is then set to evaluation and with no gradient model is tested using the test data and average BCE loss is calculated and taken as test loss.

This model's parameters and weight are saved using 'torch.save' and 'state_dict'. Encoded data is also saved as csv files for later use.

TOPIC DETECTOR

EDA & DATA PREPARATION

Data is 2235 rows of essays, description, title, source_url, author and thumbnail_url. Out of these only title and description is taken for our topic detection problem statement. Number of titles has 114 unique categories which is then reduced to improve model performance by grouping all rare categories i.e. value count less than 10 to others category.

Text in description column of this data then goes through a preprocessing function which involves making everything lower case, removing symbols, word tokenization, removing stop words, lemmatization and joining back to a single text. This process gives better usable data for deep learning.

Data is then separated as description text(x) and title(y). The description text is then vectorized at word level using keras tokenizer and padded to a length of 50. Title are encoded using label encoder. Data is then made ready for training using torch Dataset and Dataloader.

MODEL TRAINING AND EVALUATION

Loss function for this model is selected as **CrossEntropyloss** along with class weights calculated using sklearn class weights. Optimizer used for this model is **Adam** with a learning rate of 0.001. For evaluation precision and recall scores were used.

Neural network is created with 3 hidden layers they are,

- **Embedding** with 10000 embeddings and dimension of 50.
- **LSTM** with input of 50, hidden size of 32 and 2 layers.
- **Linear layer** of input 32 and 76(number of classes) output.

Model is set to train optimizer set to zero grad and the model is trained with an epoch of 20. Model is then set to evaluation and with no gradient model is tested.

This model's parameter weights are then saved using state dict and torch save. Tokenizer and label encoder were also saved using pickle for future use.

DIGIT RECOGNIZER

DATA PREPARATION

Mnist data is used for this model. This data has columns of pixels of gray scale images. Here each row is an image and each column is a pixel. There were a total of 60000 rows and each row had a label column and 784(28x28) columns. Distribution of labels is checked using countplot.

To make this data suitable to predict color data also the data in this dataset is colorized at random. Transform to convert the image to usable data for image processing is defined. A class of Image dataset for this type of data is created based on torch Dataset. Data is set ready using Dataloader.

MODEL TRAINING AND EVALUATION

Loss function used for this model is **CrossEntropyloss** along with Optimizer used for this model is **Adam** with a learning rate of 0.001. Precision and Recall scores are used for evaluating this model.

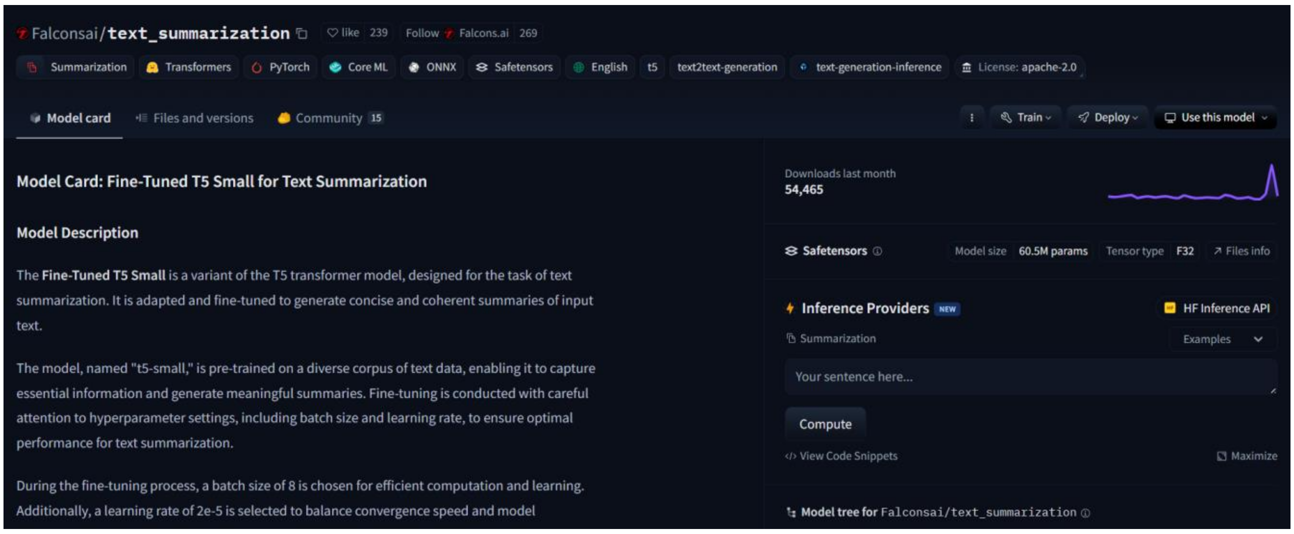
I have segregated this model's neural network into 2. First feature extraction with 2 convolution layers, 2 ELU activation function after each layer and a flatten function to flatten the features. Next classification this part has 2 simple linear layers with an output of number of classes i.e. number of digits 10. Data goes through feature extractor first and then through classifier in forward.

Model is set to train optimizer set to zero grad and the model is trained with an epoch of 30. Model is then set to evaluation and with no gradient model is tested using test data.

This model's parameter weights are then saved using state dict and torch save.

TOPIC SUMMARIZER

For this problem statement it was mentioned to use a pretrained LLM model from hugging face or GPT. So **Falconsai/text_summarization** model from hugging face was loaded as it is a suitable model for text summarization and is lesser in size compared to other models. This saved using pickle for further ease of use.



Falconsai/**text_summarization** like 239 Follow Falcons.ai 269

Summarization · Transformers · PyTorch · Core ML · ONNX · Safetensors · English · t5 · text2text-generation · text-generation-inference · License: apache-2.0

Model card · Files and versions · Community 15 · Train · Deploy · Use this model

Model Card: Fine-Tuned T5 Small for Text Summarization

Model Description

The **Fine-Tuned T5 Small** is a variant of the T5 transformer model, designed for the task of text summarization. It is adapted and fine-tuned to generate concise and coherent summaries of input text.

The model, named "t5-small," is pre-trained on a diverse corpus of text data, enabling it to capture essential information and generate meaningful summaries. Fine-tuning is conducted with careful attention to hyperparameter settings, including batch size and learning rate, to ensure optimal performance for text summarization.

During the fine-tuning process, a batch size of 8 is chosen for efficient computation and learning. Additionally, a learning rate of 2e-5 is selected to balance convergence speed and model

Downloads last month
54,465

Safetensors · Model size **60.5M params** · Tensor type **F32** · Files info

Inference Providers NEW · HF Inference API

Summarization · Examples

Your sentence here...

Compute

View Code Snippets · Maximize

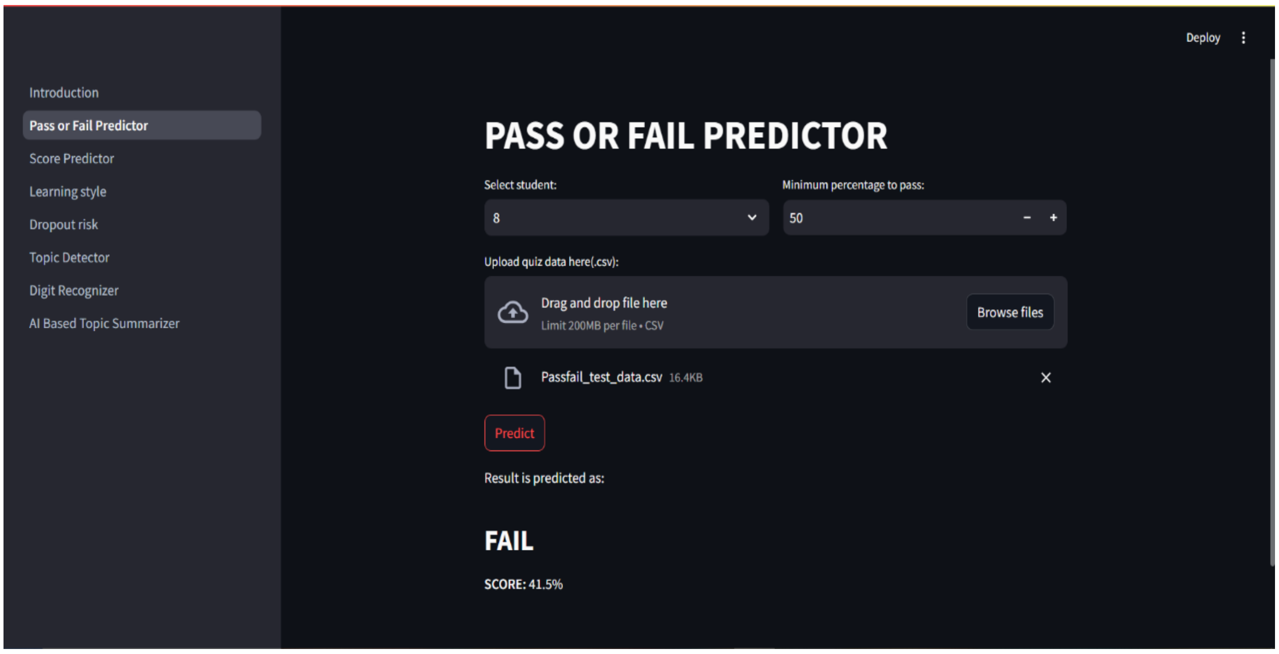
Model tree for Falconsai/text_summarization

Hugging face page of used model

EXECUTION

All the models are included as pages in an interactive web app created using **streamlit.** This app has 8 pages with the first page being Introduction which provides a brief explanation of the problem statement and pages in the app.

Pass-fail predictor when provided test details as csv file input and selecting the student ID predicts whether the student will pass or fail the given test/quiz. There is an option to change the pass percentage for the test also.



Select student: 8

Minimum percentage to pass: 50

Upload quiz data here(.csv):

Passfail_test_data.csv 16.4KB

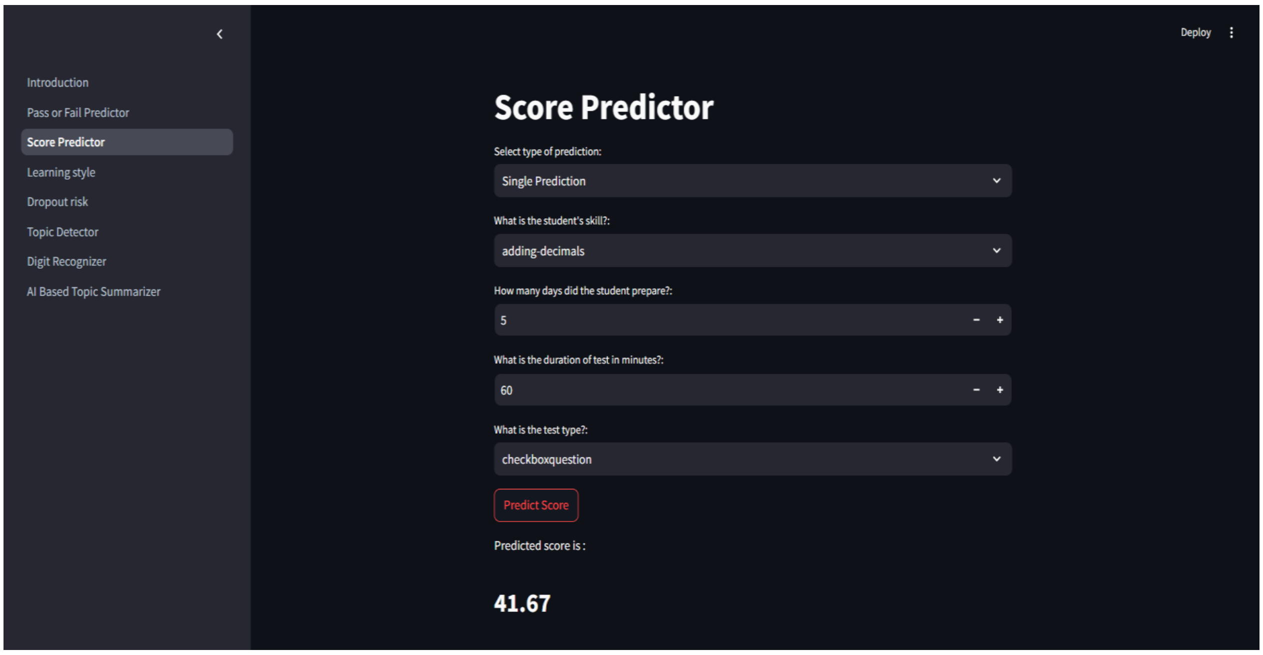
Result is predicted as:

FAIL

SCORE: 41.5%

Pass fail predictor

Score predictor comes with two options one is single where you can provide data about a single student and predict his score in his next test. Multiple here you can upload a csv file of student data for which the scores of students will be predicted and displayed. This provides an option to download for storing this table in your pc.



Score Predictor

Select type of prediction: Single Prediction

What is the student's skill?: adding-decimals

How many days did the student prepare?: 5

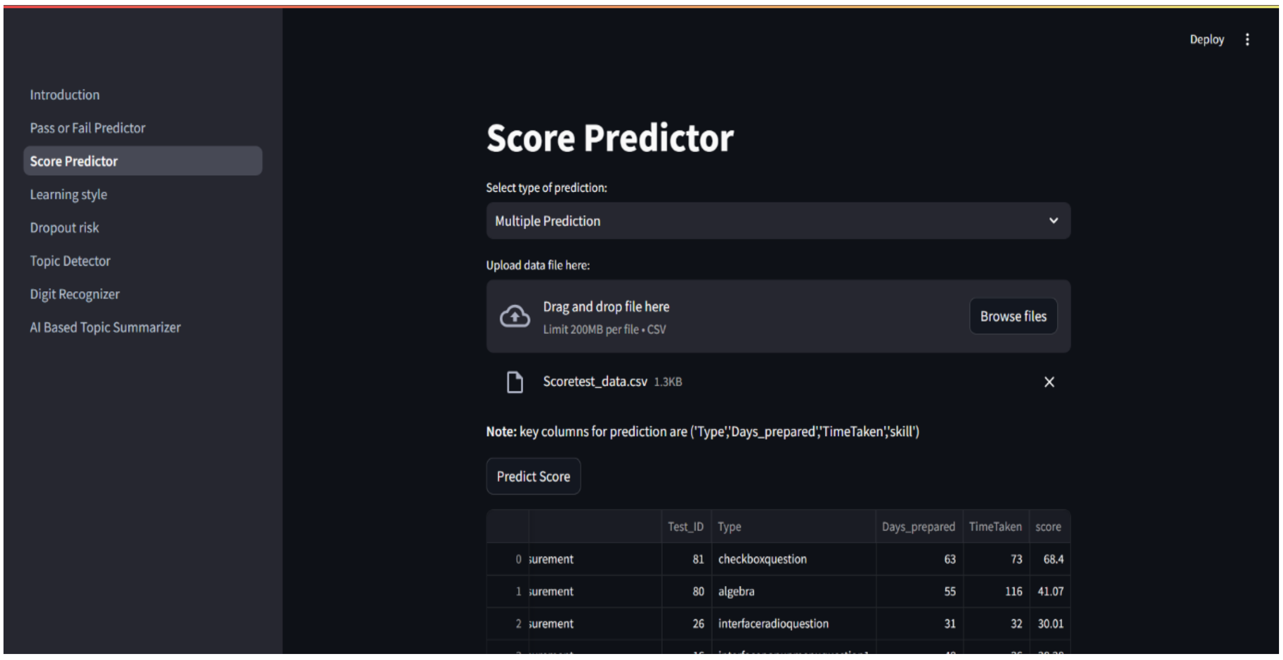
What is the duration of test in minutes?: 60

What is the test type?: checkboxquestion

Predicted score is :

41.67

Score Predictor (Single)



Score Predictor

Select type of prediction: Multiple Prediction

Upload data file here:

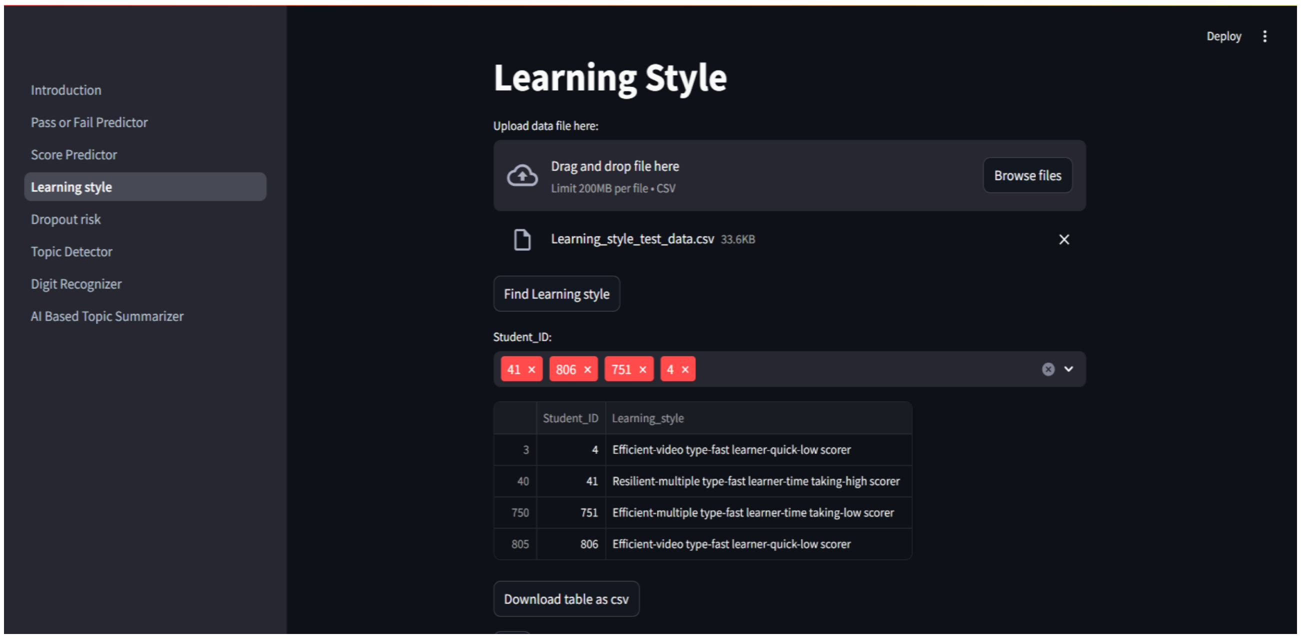
Scoretest_data.csv 1.3KB

Note: key columns for prediction are ('Type','Days_prepared','TimeTaken','skill')

		Test_ID	Type	Days_prepared	TimeTaken	score
0	surement	81	checkboxquestion	63	73	68.4
1	surement	80	algebra	55	116	41.07
2	surement	26	interfaceradioquestion	31	32	30.01

Score Predictor (Multiple)

Learning style takes student data as a csv file in input and predicts learning styles of all student in it. It gives student IDs and their learning styles as output. There is an option to filter students based on their student ID in output table. Provision for downloading output is available at bottom of the table.



Learning Style

Upload data file here:

Drag and drop file here
Limit 200MB per file • CSV

Browse files

Learning_style_test_data.csv 33.6KB

Find Learning style

Student_ID:

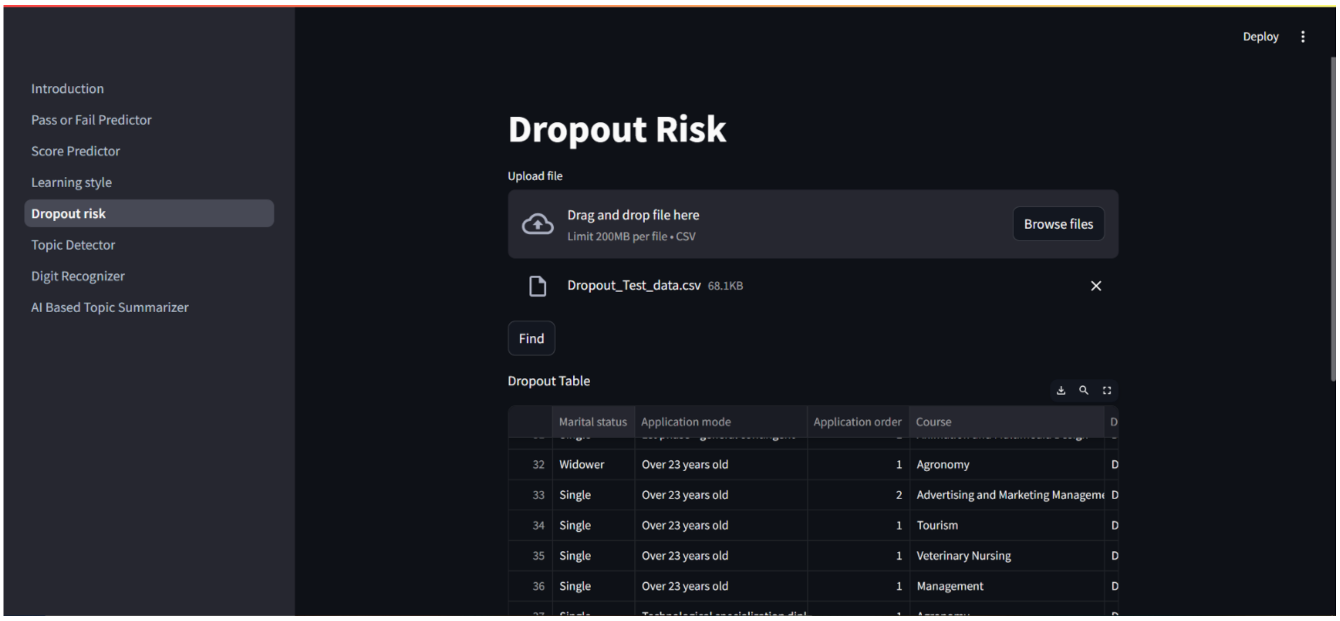
41 × 806 × 751 × 4 ×

	Student_ID	Learning_style
3	4	Efficient-video type-fast learner-quick-low scorer
40	41	Resilient-multiple type-fast learner-time taking-high scorer
750	751	Efficient-multiple type-fast learner-time taking-low scorer
805	806	Efficient-video type-fast learner-quick-low scorer

Download table as csv

Learning style predictor

Dropout risk detector takes csv file of student data and detects the risk of drop out for every student. Output is a table of students who have a risk of drop out. There are 2 download option one is to download the dropout table that is the table of students with the risk of dropout. Another is the original table provided with dropout column added in it.



Dropout Risk

Upload file

Drag and drop file here
Limit 200MB per file • CSV

Browse files

Dropout_Test_data.csv 68.1KB

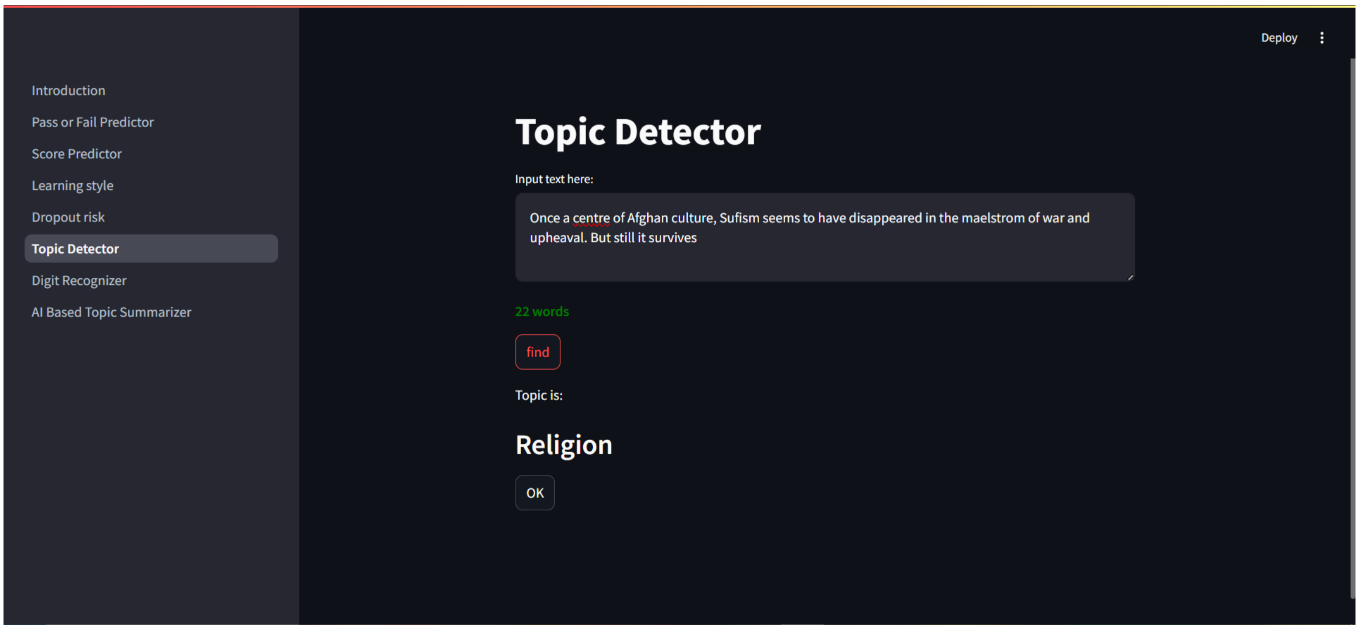
Find

Dropout Table

	Marital status	Application mode	Application order	Course	D
32	Widower	Over 23 years old	1	Agronomy	D
33	Single	Over 23 years old	2	Advertising and Marketing Manageme	D
34	Single	Over 23 years old	1	Tourism	D
35	Single	Over 23 years old	1	Veterinary Nursing	D
36	Single	Over 23 years old	1	Management	D

Dropout risk Detector

Topic Detector has a text box for the text for topic detection. Input the text in this text box and give find the app provides topic of the text as result.



Topic Detector

Input text here:

Once a centre of Afghan culture, Sufism seems to have disappeared in the maelstrom of war and upheaval. But still it survives

22 words

find

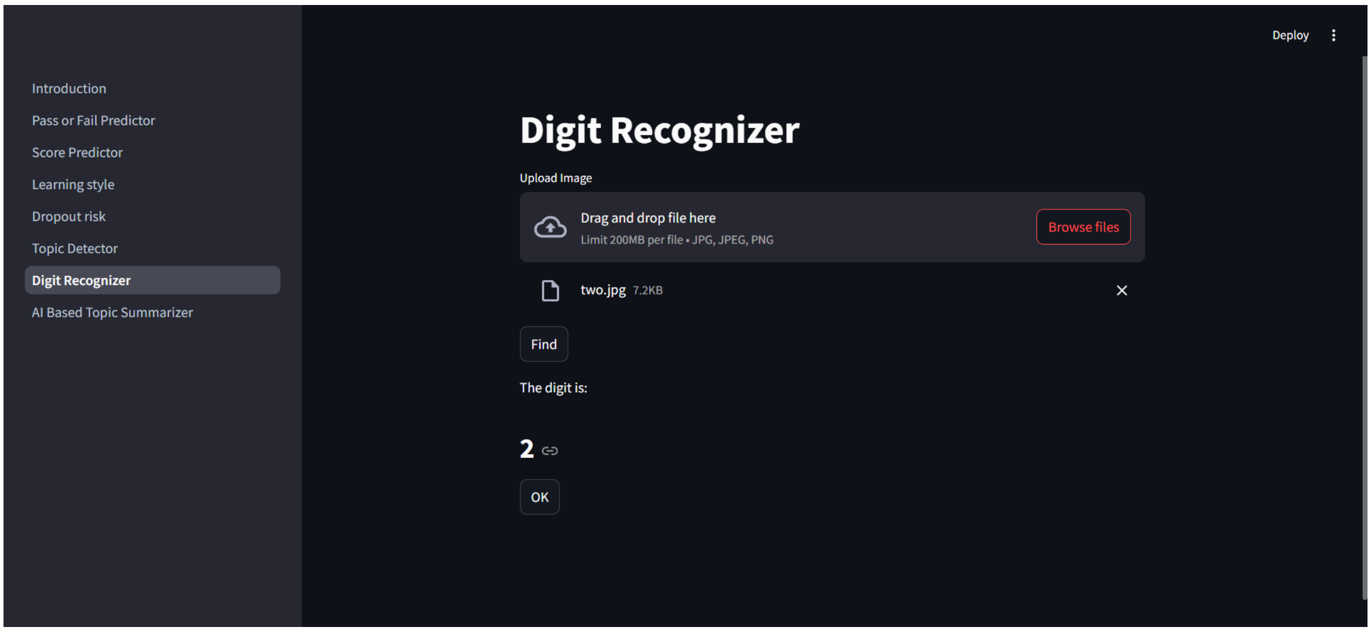
Topic is:

Religion

OK

Topic Detector

Digit Recognizer uses images files given through file uploader to find and give the number in the image as output.



Digit Recognizer

Upload Image

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

two.jpg 7.2KB

Find

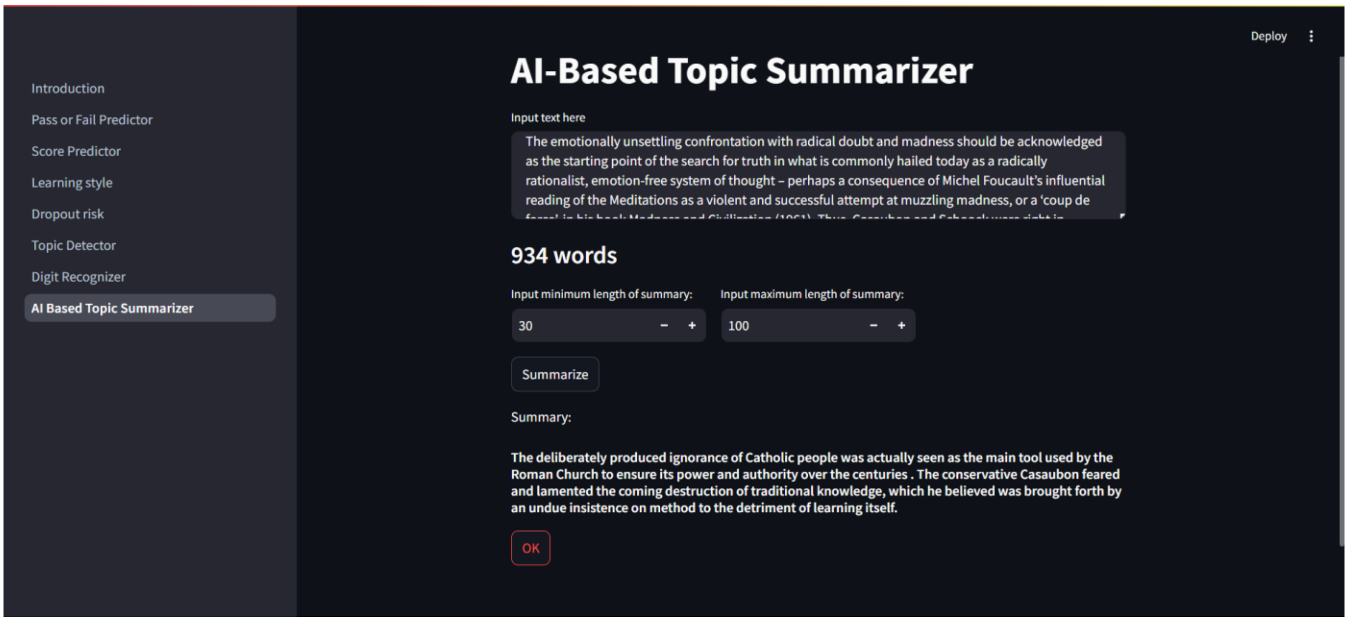
The digit is:

2

OK

Digit Recognizer

Topic summarizer has text box which takes in essays as text input and summarizes it. There is provision to also select the minimum and maximum length of summarization.



AI-Based Topic Summarizer

Input text here

The emotionally unsettling confrontation with radical doubt and madness should be acknowledged as the starting point of the search for truth in what is commonly hailed today as a radically rationalist, emotion-free system of thought – perhaps a consequence of Michel Foucault's influential reading of the Meditations as a violent and successful attempt at muzzling madness, or a 'coup de

934 words

Input minimum length of summary: 30

Input maximum length of summary: 100

Summarize

Summary:

The deliberately produced ignorance of Catholic people was actually seen as the main tool used by the Roman Church to ensure its power and authority over the centuries . The conservative Casaubon feared and lamented the coming destruction of traditional knowledge, which he believed was brought forth by an undue insistence on method to the detriment of learning itself.

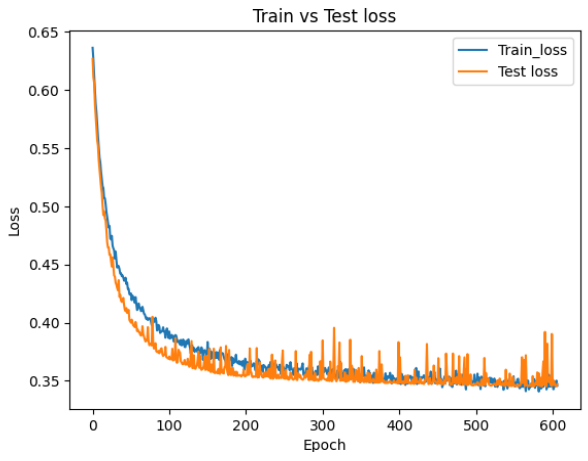
OK

Topic Summarizer

CHALLENGES FACED & FUTURE IMPROVEMENTS

CHALLENGES FACED

- Data usually has many unwanted columns and the required columns in different forms. Unwanted columns needed to be identified individually and removed. Columns in different forms needed to be transformed to create a new column that is required. This requires knowledge on the subject discussed in project.
- Normal SVC with linear kernel was giving bad scores and was slow in training. For which I had to switch to Linear which is a SVC library optimized for linear cases. It provides better performance and scores.
- In score predictor during EDA, performance for each student was different and while visualizing the relation between score and time taken for each student chart was so congested and not readable. I had to use separate chart for each student resulting in lots of charts.
- During clustering the data was so closely packed together, so while forming clusters silhouette score was high with only 2 clusters, if the number of clusters is increasing the silhouette score keeps dropping. Also, with clusters formed in this condition I could not form meaningful groups. I created groups first based on the columns provided and checked for number of groups. I used this number of clusters in KMeans algorithm.
- During binary classification Deep learning for dropout risk detection model was starting to over fit at some point. To solve it I included an early stopping procedure if the model's loss keeps increasing continuously for multiple times. I also made a table of epoch and loss and made a visualization to check where the lowest point where training score and validation score meet and consider this number of epochs for model training for best results.



Visualization of train and test loss for binary classification

- Training data on gray scale images reduced accuracy when using colored images so I had to augment RGB in the mnist dataset using random r, g, b values to train my model to work with color images as well.
- Using file path in one computer causes the issues while loading the python notebook in another computer. Using relative path to the folder is not guaranteed to always work. To solve this, I used Path from **pathlib** library to save the path from the root drive to the project file to a variable ROOT which is called every time a file path is mentioned.

FUTURE ENHANCEMENTS

Future enhancement scopes for this project are,

- Pass fail prediction can be done for multiple students and multiple tests at a time similar to multiple option available for score prediction to make it efficient.
- Topic detector can be improved to take longer texts and form new topics from the text itself instead of classification.
- Digit recognizer can be improved to detect multiple digits at same time making it to identify more than one-digit numbers. This needs the processing to not only detect there are two digits and to understand gapping between them.
- Many apps require specific columns to be available in the data to give a prediction. This needs to be updated to take in similar column names as a feature if the feature matches the required feature.