Configuration Manual: Enhancing Machine Learning Performance using Feature Engineering Techniques for Online Course Recommendation System

MSc Research Project
Data Analytics

Srivatsav Kattukottai Mani
Student ID: x18145922

School of Computing
National College of Ireland

Supervisor: Dr. Anh Duong Trinh

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Srivatsav Kattukottai Mani
Student ID:	x18145922
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr. Anh Duong Trinh
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual: Enhancing Machine Learning Performance using Feature Engineering Techniques for Online Course Recommendation System
Word Count:	780
Page Count:	16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	13th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual: Enhancing Machine Learning Performance using Feature Engineering Techniques for Online Course Recommendation System

Srivatsav Kattukottai Mani
x18145922

1 Introduction

This Configuration Manual lists together all prerequisites needed to reproduce the research project "Enhancing Machine Learning Performance using Feature Engineering Techniques for Online Course Recommendation System".
This report is organized as follows,

- Environment configuration provided in Section 2.
- Information about data gathering is detailed in Section 3.
- Data pre-processing including EDA are included in Section 4.
- Data transforming techniques implemented are detailed in Section 5.
- Details about ML models that were implemented are provided in Section 6.
- Evaluation is explained in Section 7.

2 Environment Specification

Both the Hardware and Software configurations are detailed below.

2.1 Hardware Configuration

Hardware configurations of the system are as below Figure 1

	Hardware Configuration
System	Lenovo ThinkPad X390
OS	Windows 10 Enterprise - 64-bit OS
Processor	Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz
RAM	16.0 GB (15.8 GB usable)
Hard Disk	235GB SDD
Graphics Card	Intel(R) UHD Graphics 620

Figure 1: Hardware configuration

2.2 Software Configuration

Software configurations of the system are as below Figure 2 Anaconda Navigator shown in Figure 3 can be downloaded and installed from `https://www.anaconda.com/download`. Python for windows can be downloaded and installed from `https://www.python.org/downloads/windows/`. For visualization, MS Excel is used.

	Software Configuration
OS	Windows 10 Enterprise - 64-bit OS
Anaconda	Anaconda Navigator 2.4.2
Jupyter Notebook	Notebook 6.5.4
Python version	Python 3.11.3

Figure 2: Software configuration

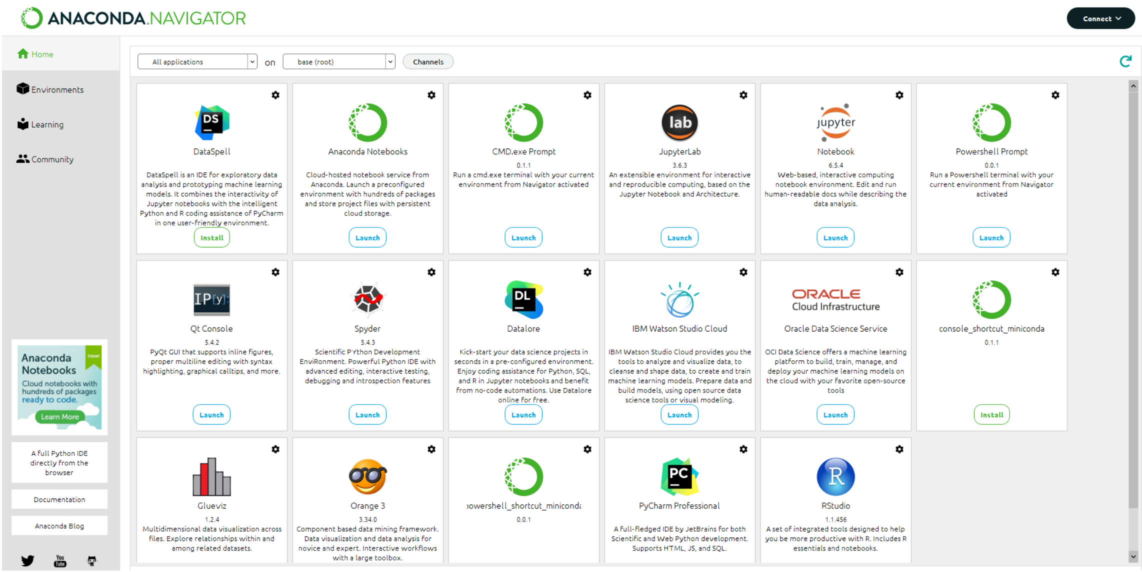

Figure 3: Anaconda Navigator

The code can be run in Jupyter notebook. This will open Jupyter notebook in web browser. The web browser will show the folder structure of the system, move to the folder where the code file and dataset is located. Open the code file from the folder and to run the code, go to Kernel menu and click Restart and Run All.

3 Data Collection

The dataset is taken from Kaggle which is a public repository as shown in Figure 4. Url of the dataset used in this research: `https://www.kaggle.com/datasets/khusheekapoor/udacity-courses-dataset-2021`. The dataset has details about online courses available in Udacity during 2021.

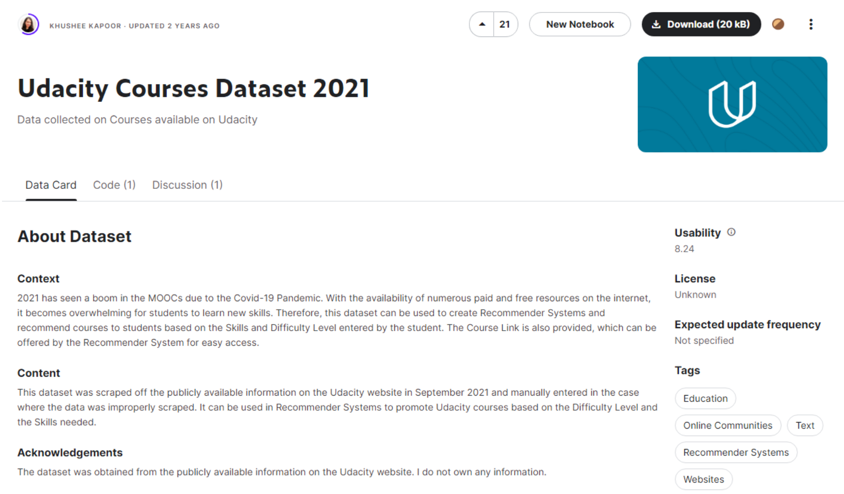

Figure 4: Udacity dataset form Kaggle

4 Data Exploration

Figure 5 includes a list of every Python library necessary to complete the project.

```
#installing all libraries
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import nltk
import string
import re
from nltk.corpus import stopwords
import spacy, gensim

import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import heatmap
from scipy.stats import mode
from numpy.linalg import norm
from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, f1_score,recall_score
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from tqdm import tqdm
```

Figure 5: Importing necessary Python Libraries

Figure 6 and 7 represents the block of code to import and read the data information.

```
udacity = pd.read_csv('Udacity.csv')
udacity
```

	Name	School	Difficulty Level	Rating	Link	About
0	Data Engineer	School of Data Science	Intermediate	4.6	https://www.udacity.com//course/data-engineer-...	Data Engineering is the foundation for the new...
1	Data Scientist	School of Data Science	Advanced	4.7	https://www.udacity.com//course/data-scientist...	Build effective machine learning models, run d...
2	Data Analyst	School of Data Science	Intermediate	4.6	https://www.udacity.com//course/data-analyst-n...	Use Python, SQL, and statistics to uncover ins...
3	C++	School of Autonomous Systems	Intermediate	4.6	https://www.udacity.com//course/c-plus-plus-na...	Get hands-on experience by building five real-...
4	Product Manager	School of Product Management	Beginner	4.7	https://www.udacity.com//course/product-manage...	Envision and execute the development of indust...
...	...	...	...	...	...	...
258	Front-End Interview Prep	Career Advancement	Intermediate	None	https://www.udacity.com//course/front-end-inte...	Answer front-end technical and behavioral inte...
259	Full-Stack Interview Prep	Career Advancement	Intermediate	None	https://www.udacity.com//course/full-stack-int...	Answer common full stack and web security inte...
260	Data Structures & Algorithms in Swift	Career Advancement	Intermediate	None	https://www.udacity.com//course/data-structure...	Review and practice the skills technical inter...
261	iOS Interview Prep	Career Advancement	Intermediate	None	https://www.udacity.com//course/ios-interview-...	Answer iOS and mobile development interview qu...
262	VR Interview Prep	Career Advancement	Intermediate	None	https://www.udacity.com//course/vr-interview-p...	Learn how to tackle interview questions for te...

263 rows × 6 columns

Figure 6: Importing Udacity dataset

```
udacity.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Name              263 non-null    object
 1   School            263 non-null    object
 2   Difficulty Level  259 non-null    object
 3   Rating            263 non-null    object
 4   Link              263 non-null    object
 5   About             255 non-null    object
dtypes: object(6)
memory usage: 12.5+ KB
```

Figure 7: EDA for checking Udacity info

4.1 Data cleaning

Udacity dataset is assigned to variable ‘data’ and Figure 8 and 9 shows the cleaning steps are performed to renaming the feature column from School to University and removing null values.

```
udacity.rename(columns = {'School':'University'}, inplace = True)
udacity['Course']=udacity['Name']+' '+udacity['About']
udacity = udacity[['Course', 'University']]
udacity
```

	Course	University
0	Data Engineer Data Engineering is the foundati...	School of Data Science
1	Data Scientist Build effective machine learnin...	School of Data Science
2	Data Analyst Use Python, SQL, and statistics t...	School of Data Science
3	C++ Get hands-on experience by building five r...	School of Autonomous Systems
4	Product Manager Envision and execute the devel...	School of Product Management
...	...	...
258	Front-End Interview Prep Answer front-end tech...	Career Advancement
259	Full-Stack Interview Prep Answer common full s...	Career Advancement
260	Data Structures & Algorithms in Swift Review a...	Career Advancement
261	iOS Interview Prep Answer iOS and mobile devel...	Career Advancement
262	VR Interview Prep Learn how to tackle intervie...	Career Advancement

263 rows × 2 columns

Figure 8: Renaming Column School to University

```
data = udacity
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Course      255 non-null    object
 1   University  263 non-null    object
dtypes: object(2)
memory usage: 4.2+ KB
```

```
data.head()
```

	Course	University
0	Data Engineer Data Engineering is the foundati...	School of Data Science
1	Data Scientist Build effective machine learnin...	School of Data Science
2	Data Analyst Use Python, SQL, and statistics t...	School of Data Science
3	C++ Get hands-on experience by building five r...	School of Autonomous Systems
4	Product Manager Envision and execute the devel...	School of Product Management

```
data.isnull().sum()
```

```
Course        8
University    0
dtype: int64
```

```
data.dropna(inplace=True)
```

Figure 9: Checking and dropping null values

5 Data Transformation

Figure 10 and 11 are functions to remove stopwords, clear the punctuations and de-contract the words from the course details.

```python
stop = set(stopwords.words('english')) #set of stopwords
sno = nltk.stem.SnowballStemmer('english') #initialising the snowball stemmer

#Stemmer reduce a word to its word stem that affixes to suffixes and prefixes or to the root of word known as lemma.

def cleanpunc(sentence): #function to clean the word of any punctuation or special characters
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    return  cleaned

print("Stopwords:",stop)
```

```
Stopwords: {'to', 'she', 'off', 'between', 'so', 'until', 'own', 'where', 'the', "mightn't", 'haven', 'has', "hasn't", 'does', '
'over', 'we', 'mustn', "wouldn't", "didn't", 'his', 'up', 'their', 'ain', 'him', 'isn', 'then', 'wouldn', 'for', 'yourself', 'w:
'with', 'o', 'just', 'our', "shouldn't", 'as', 'he', 'me', 'there', 'mightn', "shan't", 'once', 'himself', 'no', 'too', 'from',
'below', 'whom', 'should', "weren't", "isn't", 'they', 'which', 't', 've', 'when', 'ma', 'her', "haven't", 'not', 'during', 'in
oth', 'my', "she's", 'more', 'wasn', 'this', 'yours', 'of', 'some', 'only', 'shan', 'them', 'd', 'few', "won't", 'all', 'hadn',
r', 'had', "you'd", 'is', 'did', 'weren', "you're", 'its', 'most', 'these', 'do', 'you', 'under', 'out', 'nor', 'against', 'aga:
```

Figure 10: functions to remove stopwords and punctuations

```python
def decontracted(phrase):
    # This function decontract words like it's to it is.

    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)

    return phrase
```

Figure 11: functions to de-contract the words

Figure 12 illustrate the code to clean the course details, each word, de-contracted and cleaned. Figure 13 shows the course data before and after cleaning.

```python
# Here we are cleaning the data using functions define above, removing stopword and reducing words to there root words.

i=0
str1=''
final_string=[]

for sent in tqdm(data.Course.values):
    filtered_sentence=[]
    sent=cleanpunc(sent)
    sent = decontracted(sent)
    sent = sent.replace('XXXX', " ")
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(sno.stem(cleaned_words.lower())).encode('utf8')
                    filtered_sentence.append(s)
                else:
                    continue
            else:
                continue

    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    final_string.append(str1)
    i+=1

data["CleanCourse"] = final_string
data['CleanCourse']=data['CleanCourse'].str.decode("utf-8")
```

```
100%|██████████| 255/255 [00:00<00:00, 1220.08it/s]
```

Figure 12: Cleaning the course details

```python
data.tail()
```

	Course	University	CleanCourse
258	Front-End Interview Prep Answer front-end tech...	Career Advancement	interview prep answer technic behavior intervi...
259	Full-Stack Interview Prep Answer common full s...	Career Advancement	interview prep answer common full stack web se...
260	Data Structures & Algorithms in Swift Review a...	Career Advancement	data structur algorithm swift review practic s...
261	iOS Interview Prep Answer iOS and mobile devel...	Career Advancement	io interview prep answer io mobil develop inte...
262	VR Interview Prep Learn how to tackle intervie...	Career Advancement	interview prep learn tackl interview question ...

Figure 13: Cleaned and Uncleaned Course details

Then, the cleaned course column is assigned to X and University is assigned to Y variables. Figures 14 shows the code used to label encode the University column and dividing the data into training and testing set. Here, 80:20 split ratio is used.

```
X=data['CleanCourse']
y=data['University']
```

```
le = LabelEncoder()
y = le.fit_transform(y)
```

```
n = int(data.shape[0] - (data.shape[0]*0.2))
n
```

```
204
```

```
X_train = X[:n]
y_train = y[:n]
X_test = X[n:]
y_test = y[n:]

X_train.shape,X_test.shape
```

```
((204,), (51,))
```

Figure 14: Label Encoding and assigning train test split

The Figure 15, illustrate the code to generate the function sent_to_words to split the cleaned course data sentences to words on both X_train and X_test splits.

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']): #'NOUN', 'ADJ', 'VERB', 'ADV'
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append(" ".join([token.lemma_ if token.lemma_ not in ['-PRON-'] else ''
                                   for token in doc if token.pos_ in allowed_postags]))
    return texts_out

data_words_train = list(sent_to_words(X_train))
print(data_words_train[:1])
```

```
[['data', 'engin', 'data', 'engin', 'foundat', 'new', 'world', 'big', 'data', 'enrol', 'build', 'data', 'infrastructur', 'essenti', 'skill', 'advanc', 'data', 'career']]
```

```
data_words_test = list(sent_to_words(X_test))
print(data_words_test[:1])
```

```
[['data', 'wrangl', 'mongodb', 'data', 'scientist', 'spend', 'time', 'clean', 'data', 'cours', 'learn', 'convert', 'manipul', 'messi', 'data', 'extract', 'need']]
```

Figure 15: Splitting sentences to words using sent_to_words

The Figure 16, illustrate the implementation of the lemmatizer function to lemmatize the course details on both train and test splits.

```
nlp = spacy.load("en_core_web_sm", disable=["parser", "ner"])
```

```
data_lemmatized_train = lemmatization(data_words_train, allowed_postags=["NOUN", "VERB"]) #select noun and verb
print(data_lemmatized_train[:2])
```

```
['datum world datum enrol build datum infrastructur skill advanc datum career', 'scientist build effect learn model recommend system deploy project']
```

```
data_lemmatized_test = lemmatization(data_words_test, allowed_postags=["NOUN", "VERB"]) #select noun and verb
print(data_lemmatized_test[:2])
```

```
['datum wrangl mongodb datum scientist spend time datum cour learn extract need', 'develop get introduct program build app']
```

Figure 16: Lemmatization

The Figure 17, illustrate the code for TFIDF vectorizer of the lemmatized training and testing data . Vectorizer creates an array for numbers for each word in the course data which can be fed into ML models.

```
vectorizer = TfidfVectorizer(analyzer='word', stop_words='english', max_features=40, lowercase=True, token_pattern='[a-zA-Z0-9]{3,}')
```

```
X_train = vectorizer.fit_transform(data_lemmatized_train).toarray()
```

```
X_test = vectorizer.fit_transform(data_lemmatized_test).toarray()
```

```
X_train.shape, X_test.shape
```

```
((204, 40), (51, 40))
```

```
print(X_train)
```

```
[[0.         0.         0.         ... 0.         0.         0.31764592]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

```
print(X_test)
```

```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.44710856 ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.40406599 0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

Figure 17: TF-IDF Vectorizer

6 ML models implemented

Here, 3 ML models as shown below are implemented on both raw and transformed course feaures and comparative study is done on the performance to prove the importance of feature engineering.

6.1 SVM on Transformed Course feature

Implementing SVM model on transformed course feature as shown in Figure 18

```
SVM =  SVC()
SVM.fit(X_train, y_train)
y_pred = SVM.predict(X_test)
SVM_accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print("Accuracy:",SVM_accuracy)
```

```
Accuracy: 68.63
```

```
SVM_precision = round(precision_score(y_test, y_pred, average="weighted"),2)
print("Precision:",SVM_precision)
```

```
Precision: 0.62
```

```
SVM_f1 = round(f1_score(y_test, y_pred, average="weighted"),2)
print("F1-score:",SVM_f1)
```

```
F1-score: 0.65
```

```
SVM_recall = round(recall_score(y_test, y_pred, average="weighted"),2)
print("Recall_score:",SVM_recall)
```

```
Recall_score: 0.69
```

Figure 18: SVM model on Transformed Course feature

6.2 KNN on Transformed Course feature

Implementing KNN model on transformed course feature as shown in Figure 19

```
KNN = KNeighborsClassifier(algorithm= 'auto', metric= 'cosine', n_neighbors= 10)
KNN.fit(X_train, y_train)
y_pred = KNN.predict(X_test)
KNN_accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print("Accuracy:",KNN_accuracy)
```

```
Accuracy: 58.82
```

```
KNN_precision = round(precision_score(y_test, y_pred, average="weighted"),2)
print("Precision:",KNN_precision)
```

```
Precision: 0.74
```

```
KNN_f1 = round(f1_score(y_test, y_pred, average="weighted"),2)
print("F1-score:",KNN_f1)
```

```
F1-score: 0.65
```

```
KNN_recall = round(recall_score(y_test, y_pred, average="weighted"),2)
print("Recall_score:",KNN_recall)
```

```
Recall_score: 0.59
```

Figure 19: KNN model on Transformed Course feature

6.3 AdaBoost on Transformed Course feature

Implementing AdaBoost model on transformed course feature as shown in Figure 20

```
adaboost = AdaBoostClassifier(n_estimators=50,learning_rate=1)
adaboost.fit(X_train, y_train)
y_pred = adaboost.predict(X_test)
adaboost_accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print("Accuracy:",adaboost_accuracy)

Accuracy: 72.55

adaboost_precision = round(precision_score(y_test, y_pred, average="weighted"),2)
print("Precision:",adaboost_precision)

Precision: 0.6

adaboost_f1 = round(f1_score(y_test, y_pred, average="weighted"),2)
print("F1-score:",adaboost_f1)

F1-score: 0.66

adaboost_recall = round(recall_score(y_test, y_pred, average="weighted"),2)
print("Recall_score:",adaboost_recall)

Recall_score: 0.73
```

Figure 20: AdaBoost model on Transformed Course feature

Assigning the uncleaned feature 'Course' to X and re-run the ML models as shown in Figure 21

```
X=data['Course']
y=data['University']
```

Figure 21: Assign 'Course' to X variable

6.4 SVM on Raw Course feature

Implementing SVM model on raw course feature as shown in Figure 22

```
SVM =  SVC()
SVM.fit(X_train, y_train)
y_pred = SVM.predict(X_test)
SVM_accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print("Accuracy:",SVM_accuracy)

Accuracy: 49.02

SVM_precision = round(precision_score(y_test, y_pred, average="weighted"),2)
print("Precision:",SVM_precision)

Precision: 0.58

SVM_f1 = round(f1_score(y_test, y_pred, average="weighted"),2)
print("F1-score:",SVM_f1)

F1-score: 0.53

SVM_recall = round(recall_score(y_test, y_pred, average="weighted"),2)
print("Recall_score:",SVM_recall)

Recall_score: 0.49
```

Figure 22: SVM model on Raw Course feature

6.5 KNN on Raw Course feature

Implementing KNN model on raw course feature as shown in Figure 23

```
KNN = KNeighborsClassifier(algorithm= 'auto', metric= 'cosine', n_neighbors= 10)
KNN.fit(X_train, y_train)
y_pred = KNN.predict(X_test)
KNN_accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print("Accuracy:",KNN_accuracy)

Accuracy: 37.25

KNN_precision = round(precision_score(y_test, y_pred, average="weighted"),2)
print("Precision:",KNN_precision)

Precision: 0.55

KNN_f1 = round(f1_score(y_test, y_pred, average="weighted"),2)
print("F1-score:",KNN_f1)

F1-score: 0.44

KNN_recall = round(recall_score(y_test, y_pred, average="weighted"),2)
print("Recall_score:",KNN_recall)

Recall_score: 0.37
```

Figure 23: KNN model on Raw Course feature

6.6 AdaBoost on Raw Course feature

Implementing AdaBoost model on raw course feature as shown in Figure 24

```
adaboost = AdaBoostClassifier(n_estimators=50,learning_rate=1)
adaboost.fit(X_train, y_train)
y_pred = adaboost.predict(X_test)
adaboost_accuracy = round(accuracy_score(y_test, y_pred)*100,2)
print("Accuracy:",adaboost_accuracy)

Accuracy: 37.25

adaboost_precision = round(precision_score(y_test, y_pred, average="weighted"),2)
print("Precision:",adaboost_precision)

Precision: 0.62

adaboost_f1 = round(f1_score(y_test, y_pred, average="weighted"),2)
print("F1-score:",adaboost_f1)

F1-score: 0.47

adaboost_recall = round(recall_score(y_test, y_pred, average="weighted"),2)
print("Recall_score:",adaboost_recall)

Recall_score: 0.37
```

Figure 24: AdaBoost model on Raw Course feature

7 Evaluation

7.1 Comparative results of ML models on Raw course data

Comparison of ML performance on raw course feature as shown in Figure 25

```
#creating comparison table for the models
from tabulate import tabulate
table = [["SVM",SVM_accuracy,SVM_precision,SVM_f1,SVM_recall],
         ["K-NearestNeighbours",KNN_accuracy,KNN_precision,KNN_f1,KNN_recall],
         ["Adaptive Boosting",adaboost_accuracy,adaboost_precision,adaboost_f1,adaboost_recall]]


print (tabulate(table, headers=["Algorithm", "Accuracy", "Precision", "F1-score", "Recall"]))
result = pd.DataFrame(table)
```

Algorithm	Accuracy	Precision	F1-score	Recall
SVM	49.02	0.58	0.53	0.49
K-NearestNeighbours	37.25	0.55	0.44	0.37
Adaptive Boosting	37.25	0.62	0.47	0.37

Figure 25: Model performance on Raw course data

7.2 Comparative results of ML models on Transformed course data

Comparison of ML performance on transformed course feature as shown in Figure 26

```
#creating comparison table for the models
from tabulate import tabulate
table = [["SVM",SVM_accuracy,SVM_precision,SVM_f1,SVM_recall],
         ["K-NearestNeighbours",KNN_accuracy,KNN_precision,KNN_f1,KNN_recall],
         ["Adaptive Boosting",adaboost_accuracy,adaboost_precision,adaboost_f1,adaboost_recall]]


print (tabulate(table, headers=["Algorithm", "Accuracy", "Precision", "F1-score", "Recall"]))
result = pd.DataFrame(table)
```

Algorithm	Accuracy	Precision	F1-score	Recall
SVM	68.63	0.62	0.65	0.69
K-NearestNeighbours	58.82	0.74	0.65	0.59
Adaptive Boosting	72.55	0.6	0.66	0.73

Figure 26: Model performance on Transformed course data

7.3 Visualization of Results

Finally, MS Excel is used to plot the below visualizations of results.

Comparison of Accuracy values for ML models as shown in Figure 27

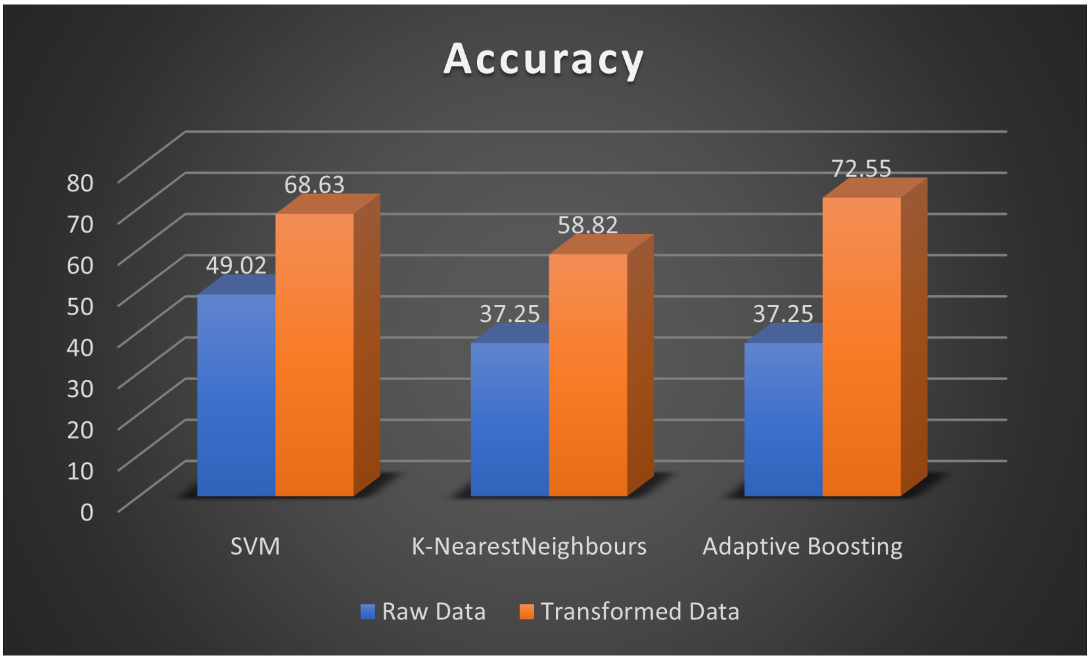

Figure 27: Accuracy

Comparison of Precision values for ML models as shown in Figure 28

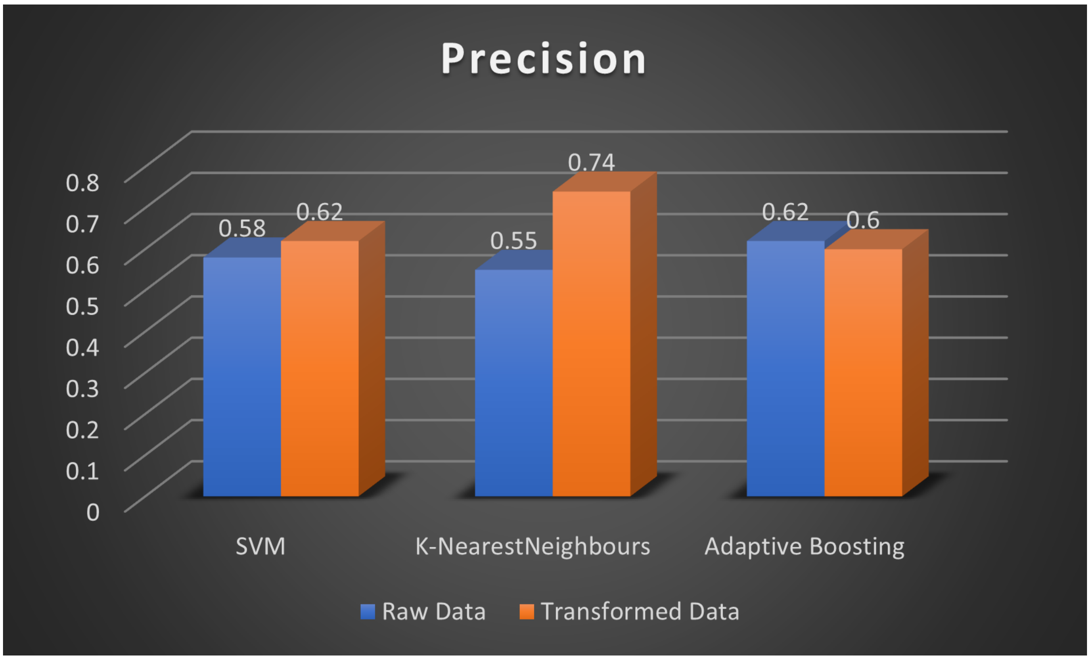

Figure 28: Precision

Comparison of F1-score values for ML models as shown in Figure 29

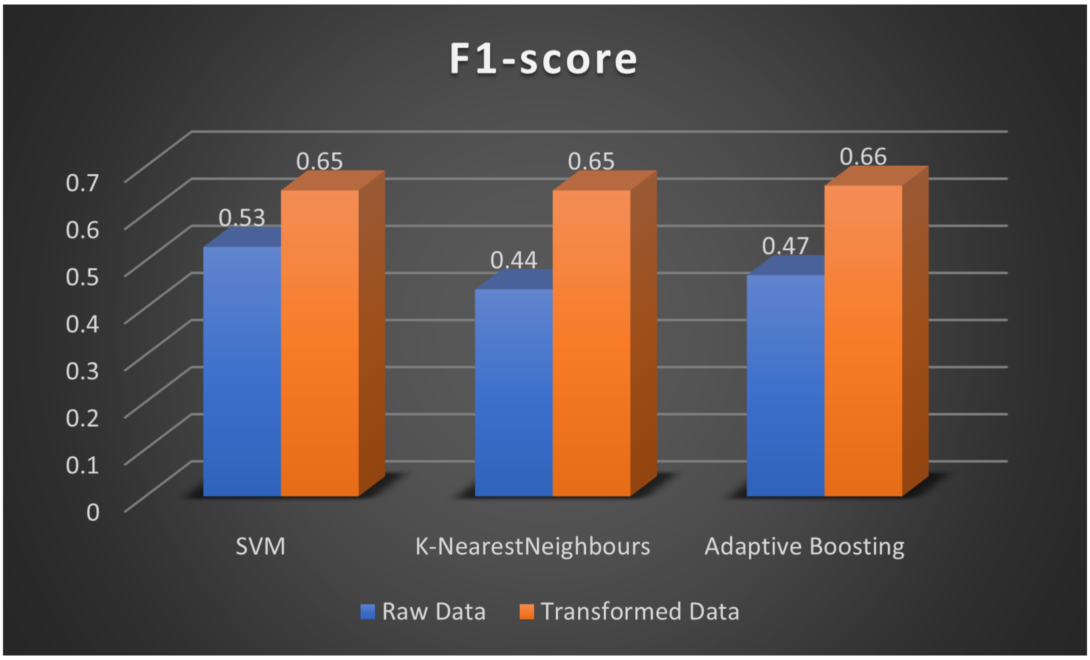

Figure 29: F1-score

Comparison of Recall score values for ML models as shown in Figure 30

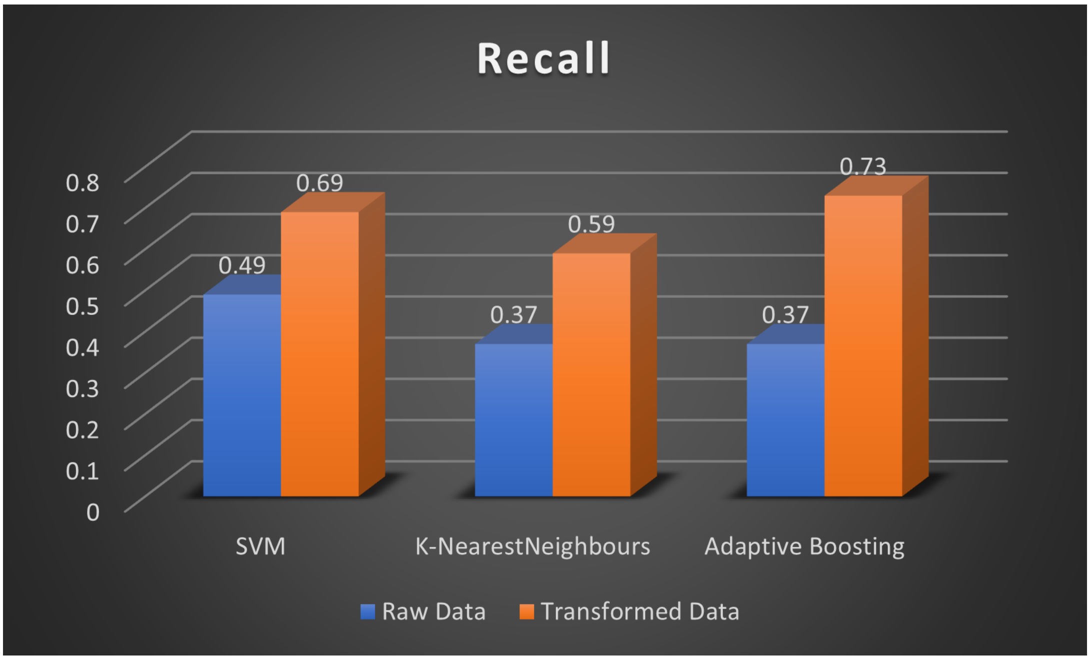

Figure 30: Recall score