Enhancing Sales of E-Learning Platforms Using Machine Learning

A Project Report To Be Submitted To Indian Institute of Technology (BHU), Varanasi In Partial Fulfilment Of The Requirements For The Award Of The Degree Of

BACHELOR OF TECHNOLOGY
IN
MECHANICAL ENGINEERING

Submitted By

KAPIL KUMAR SHARMA	17135048
PIYUSH AGRAWAL	17135061
PRIYAM GARG	17135065
SHUBHAM RAJ	17135083
VAIBHAV	17134024

Under The Esteem Guidance Of
Prof. Anil Kumar Agrawal
Department of Mechanical Engineering
Indian Institute of Technology (BHU), Varanasi

ACKNOWLEDGEMENT

It is our great fortune to have had the opportunity to work on this exciting and thought- provoking project in this institute. The learning and experience we have received here is of inexplicable value to us. Gratitude is one of the deepest expressions of one's heart. So, it gives us immense pleasure to express our paramount gratitude to each one of those who made this possible.

We would like to express our deep sense of gratitude to **Dr. Anil Kumar Agrawal, Professor, IIT (BHU)** for providing us this unique opportunity of carrying out the project and for his constant guidance, encouragement and timely support throughout the course of this project.

TABLE OF CONTENTS

INTRODUCTION

In today's world where time is a commodity in sales .One of the most critical business decisions of a company is related to customer acquisition. During the acquisition phase of the customer life cycle, companies try to convert leads into customers through different methods. One useful way to save time while increasing sales is to make sure that we focus on the best available leads and not waste time on inactive leads. In order to increase the conversion rate we used the customer specific data like time spent on a website, total visits, occupation etc. and the customers are then ranked in order of the probability of conversion , which are then pursued by sales people. In order to achieve this goal, real world data is utilized as input to various machine learning models which predicts the probability of conversion using the important features of the data.

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy. If a completely new user visits an e-commerce site, that site will not have any past history of that user. So how does the site go about recommending products to the user in such a scenario? One possible solution could be to recommend the best selling products, i.e. the products which are high in demand. Another possible solution could be to recommend the products which would bring the maximum profit to the business.

PREVIOUS WORK

Our aim of the project is to help e-learning platforms to increase their sales and customer conversion rate using machine learning. Our previous work involves data preprocessing wherein we performed data cleaning, feature scaling and one-hot encoding so that the data can be fed into machine learning models for prediction, further performing data visualization to get valuable insights and inferences from the data which can be interpreted and thus can be utilized to support better business decision making and support conclusion in order to optimize the sales. Previously we used supervised machine learning model i.e. Logistic Regression, Random Forest, Support Vector Machine, K-nearest neighbor, Naive Bayes for predicting the most potential customers.

Data Preprocessing

Data Cleaning - Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

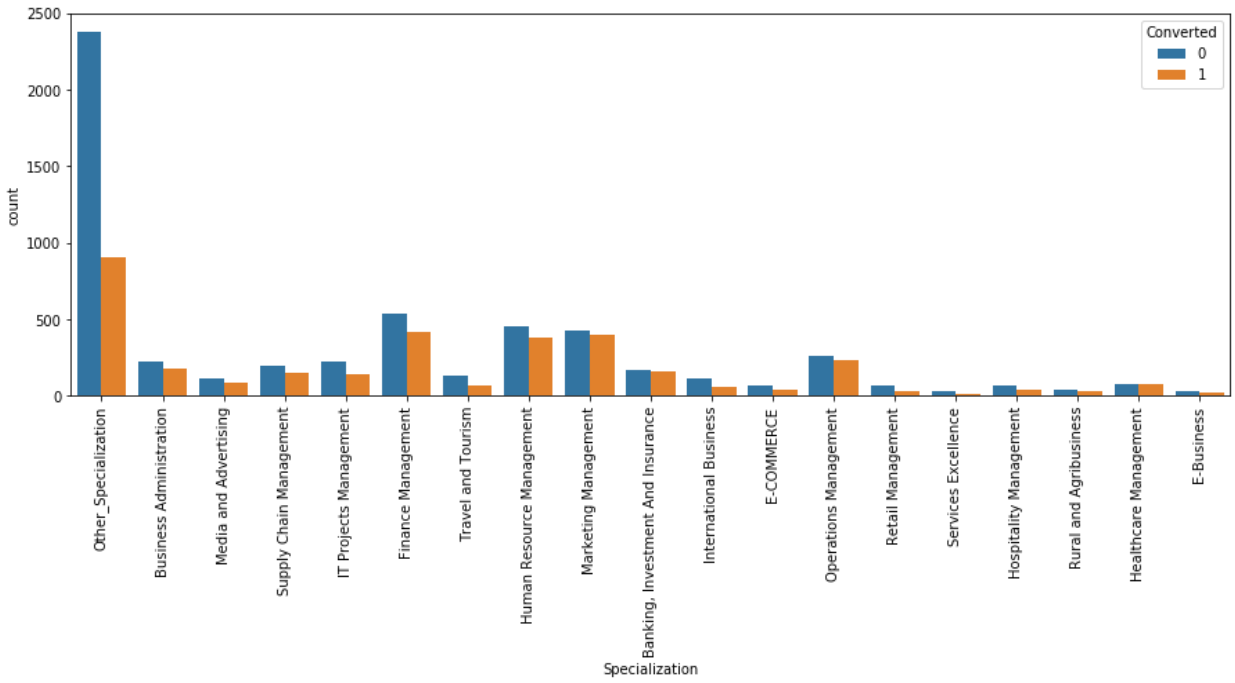
Feature Scaling - Feature scaling is a method used to normalize the range of independent variables or features of data.

One Hot Encoding - One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

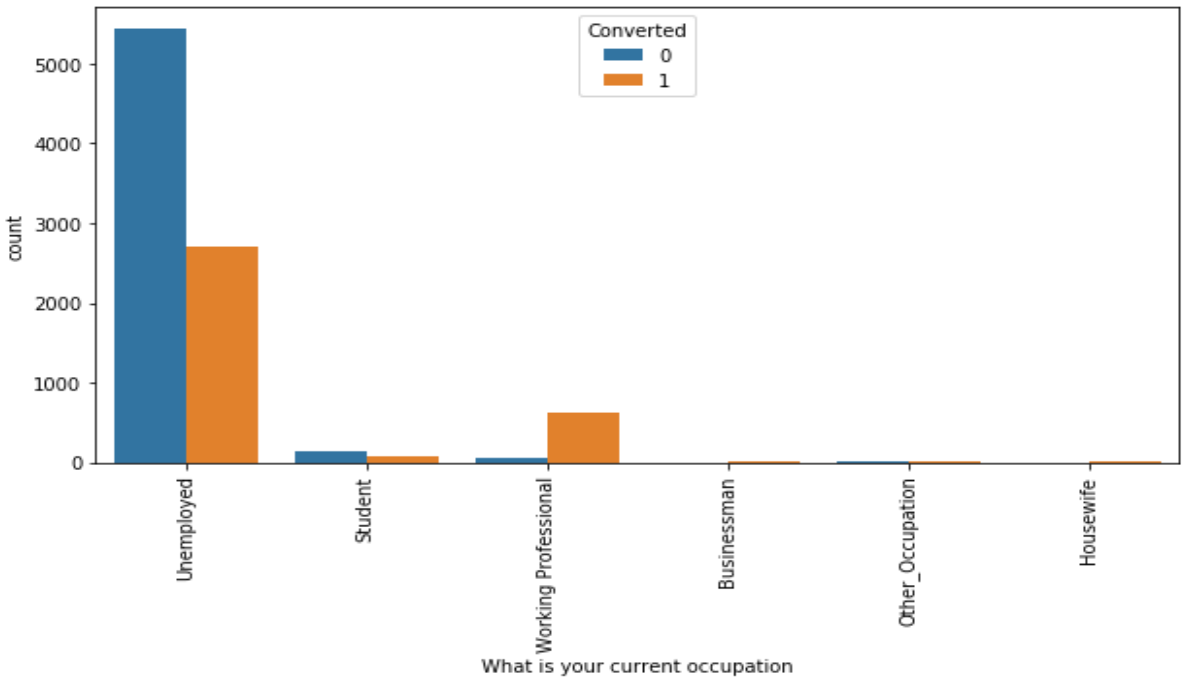
Data Visualization

Some of the data visualization of important features from our previous work:

Specialization



Occupation



Machine Learning Models

In our previous we used five different supervised learning models:

1. Logistic Regression
2. K Nearest Neighbour
3. Support Vector Machine
4. Naive Bayes
5. Random Forest Classification

Logistic Regression

Logistic regression is a predictive analysis regression technique where the dependent variable is categorical. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more independent variables. It uses a logistic function called the sigmoid function as its activation function which outputs the value between 0 and 1.

Best accurate hyperparameter tuned logistic regression model:

Optimizer	Regularization	Number of iterations	Accuracy
lbfgs	L2	200	91.36

*lbfgs- Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm

K-nearest neighbor

The principle behind nearest neighbour methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbour learning), or vary based on the local density of points (radius-based neighbour learning).

Best accurate hyperparameter tuned k-nearest neighbor model:

Metrics	Weights	Number neighbors	Accuracy
Mikowski	Uniform	5	91.06

Support Vector Machine

Support Vector Machine (SVM) is a machine learning algorithm which is be used for classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features specified) with the value of each feature being the value of a particular coordinate. Then, the algorithm performs classification by finding the hyper-plane that differentiates the two classes very well.

Best accurate hyperparameter tuned Support vector machine model:

Kernel	Gamma	C	Accuracy
Linear	Scale	1	92.37

Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable

Best accurate hyperparameter tuned Support vector machine model:

Naïve Bayes Algorithm	Accuracy
Gaussian	87.54

Random Forest

Random forest is a supervised learning algorithm which is used for classification problems. It is an ensemble tree-based learning algorithm. It is a set of decision trees from a randomly selected subset of the training set. The algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.

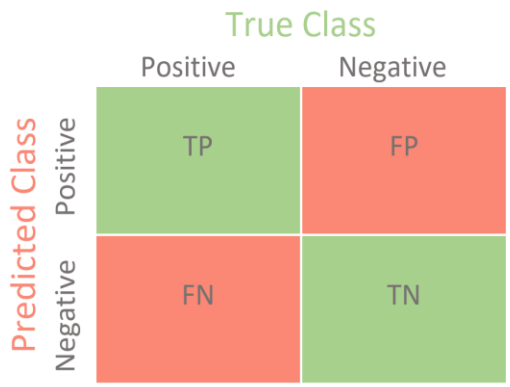
Best accurate hyperparameter tuned Support vector machine model:

Criterion of split	Max no. of features	Number estimators	Accuracy
Entropy	Auto	150	91.11

Performance Metrics Beyond Accuracy

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.



Precision

The ratio of correct positive predictions to the total predicted positives.

Precision = (TP) / (TP+FP)

Recall

The ratio of correct positive predictions to the total positives examples.

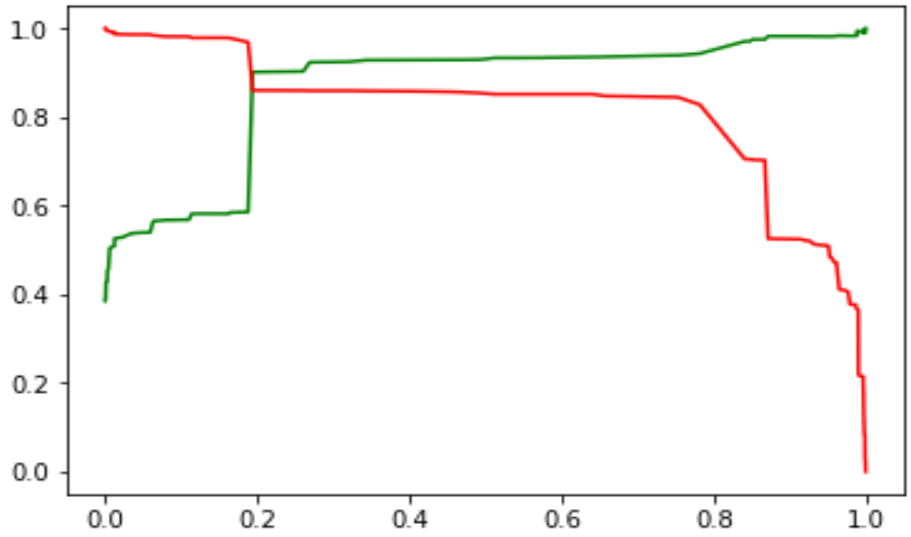
Recall = (TP) / (TP+FN)

Threshold Value

The output of a classification model is a probability. We can select a threshold value. If the probability is greater than this threshold value, the event is predicted to happen otherwise it is predicted not to happen. A confusion or classification matrix compares the actual outcomes to the predicted outcomes.

Precision-Recall Curve

A precision-recall curve (or PR Curve) is a plot of the precision and the recall for different probability thresholds.In our project we selected the optimum threshold value that gives us the best precision and recall value. The optimum threshold value comes out to be 0.2 as shown below in the precision-recall curve.



Results Obtained For Support Vector Machine Model

Accuracy- 92.37

Precision- 90.06

Recall- 86.32

FEATURE EXTRACTION

Companies have more data than ever, so it's crucial to know the difference between Useful Data and Unuseful Data. Amongst the important aspects in Machine Learning are "**Feature Selection**" and "**Feature Extraction**". Problem of selecting some subset of a learning algorithm's input variables upon which it should focus attention, while ignoring the rest. **Feature Selection** can significantly improve a learning algorithm's performance. Feature extraction involves reducing the number of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause a classification algorithm to over fit to training samples and generalize poorly to new samples.

Out of the 37 features present in the data many features have the value as "NO" as the primary answer (up to 90%), hence not much data could have been extracted from them even if we try to run the machine learning models on them. Not only these features will reduce the overall quality of the data but also degrade the model's learning ability. These features were dropped after the processing of the data.

RECURSIVE FEATURE ELIMINATION

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Recursive Feature Elimination recursively removes features, builds a model using the remaining attributes and calculates model accuracy. RFE is able to work out the combination of attributes that contribute to the prediction on the target variable Features are ranked by the model's feature_importance attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and co-linearity that may exist in the model.

RFE requires a specified number of features to keep, however it is often not known in advance how many features are required. To find the optimal number of features cross-validation is used with RFE to score different feature subsets and select the best scoring collection of features. Using RFE for each type of model all the features relevant for that specific model were selected from the dataset and the rest of the

features were dropped. Finally the data set was split randomly in the training test and test set.

The features selected by Recursive feature elimination are given below-

```
col = X_train.columns[rfe.support_]
col
```

```
Index(['Do Not Email', 'Lead Origin_Lead Add Form',
       'Lead Source_Welingak Website',
       'What is your current occupation_Working Professional', 'Tags_Busy',
       'Tags_Closed by Horizzon', 'Tags_Lost to EINS', 'Tags_Ringing',
       'Tags_Will revert after reading the email', 'Tags_invalid number',
       'Tags_switched off', 'Tags_wrong number given', 'Lead Quality_Not Sure',
       'Lead Quality_Worst', 'Last Notable Activity_SMS Sent'],
      dtype='object')
```

PRINCIPAL COMPONENT ANALYSIS

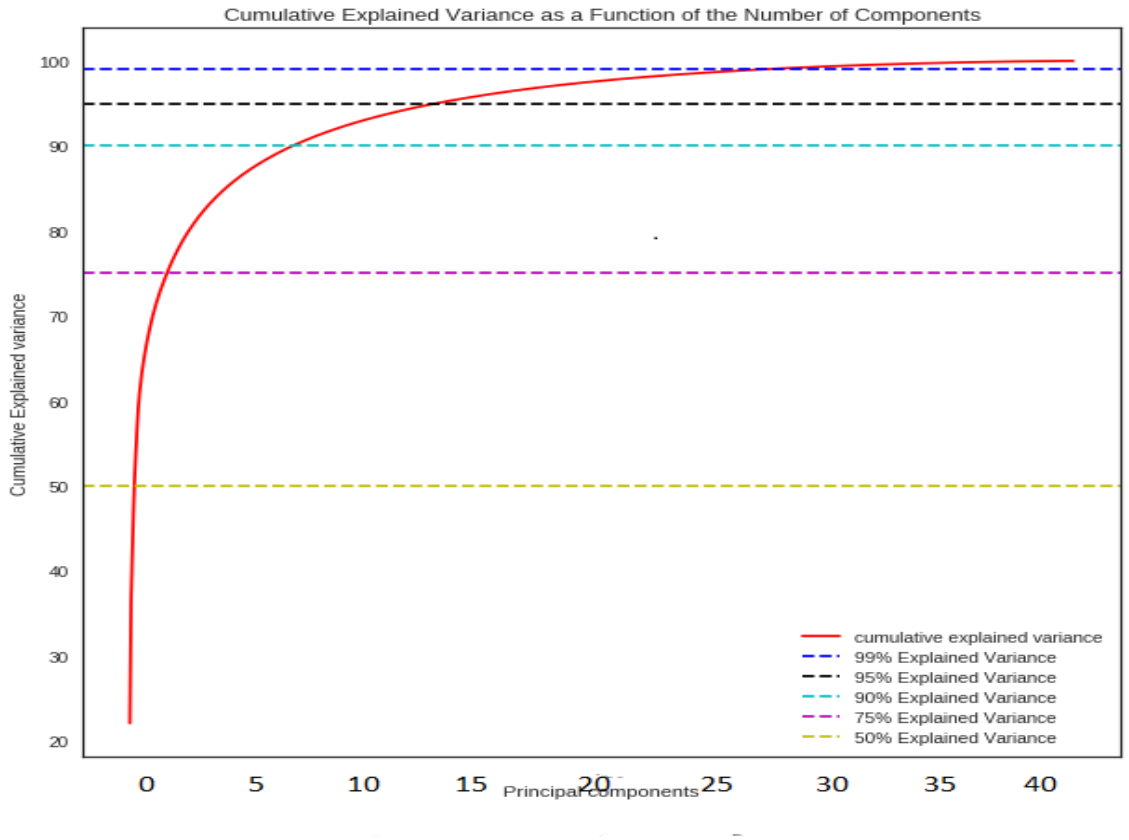
Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning .High dimensionality means that the dataset has a large number of features. The primary problem associated with high-dimensionality in the machine learning field is model overfitting, which reduces the ability to generalize beyond the examples in the training set.Models also become more efficient as the reduced feature set boosts learning rates and diminishes computation costs by removing redundant features.PCA can also be used to filter noisy datasets, such as image compression. The first principal component expresses the most amount of variance. Each additional component expresses less variance and more noise, so representing the data with a smaller subset of principal components preserves the signal and discards the noise.

PCA Workflow-

1. **Normalize the data** - PCA is used to identify the components with the maximum variance, and the contribution of each variable to a component is based on its magnitude of variance. It is best practice to normalize the data before conducting a PCA as unscaled data with different measurement units can distort the relative comparison of variance across features.

2. **Create a covariance matrix** - A useful way to get all the possible relationships between all the different dimensions is to calculate the covariance among them all and put them in a covariance matrix which represents these relationships in the data.

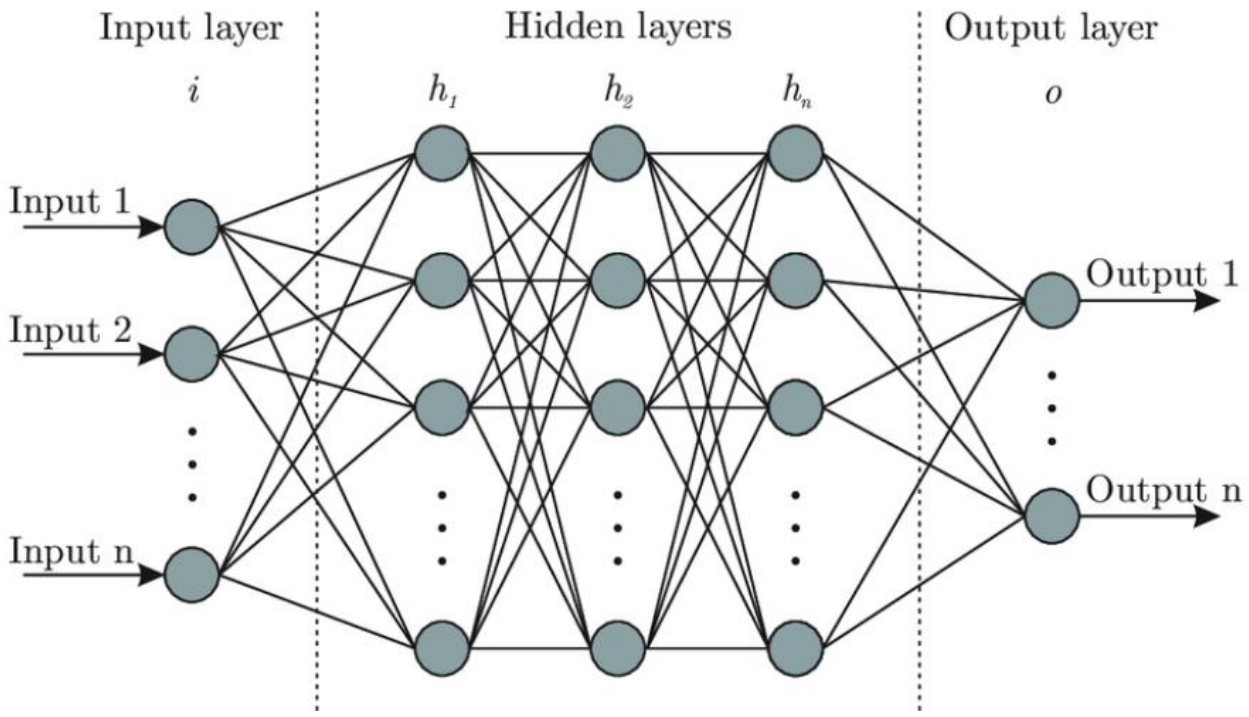
3. **Select the optimal number of principal components**-The optimal number of principal components is determined by looking at the cumulative explained variance ratio as a function of the number of components.



NEURAL NETWORK

A neural network is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form.Machine learning algorithms that use neural networks generally do not need to be programmed with specific rules that define what to expect from the input. The neural net learning algorithm instead learns from processing many labeled examples that are supplied during training and using this answer key to learn what characteristics of the input are needed to construct the correct output. Once a sufficient number of examples have been processed, the neural network can begin to process new, unseen inputs and successfully return accurate results.

The architecture of neural networks is as follows:-



1. The input layer, which takes training values as inputs.
2. Hiddens layers which tries to establish the hidden relation between input and output variables.
3. Output layer which provides the final prediction of the model.

Hyperparameters

Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model being trained.

Needs of hyperparameter tuning:-

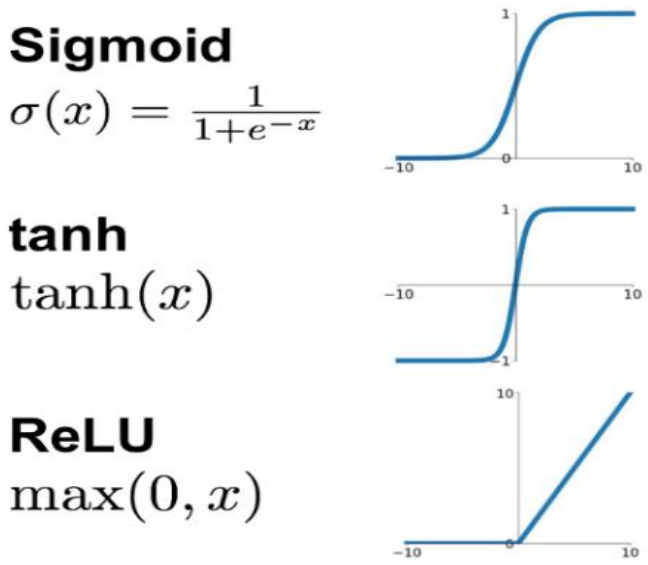
1. To find the right balance between bias and variance
2. To prevent the model from falling into vanishing/exploding gradient problem
3. Encountering local optima.
4. Prevent the no convergence of the model.

Hyperparameter tuning done in our models:-

1. **Number of Layers**:- It was chosen in accordance with the thought that a very high number may introduce problems like overfitting and vanishing and exploding gradient problems and a lower number may cause a model to have high bias and low potential model. The number of hidden layers during hyperparameter tuning were in the range of 2 to 5.

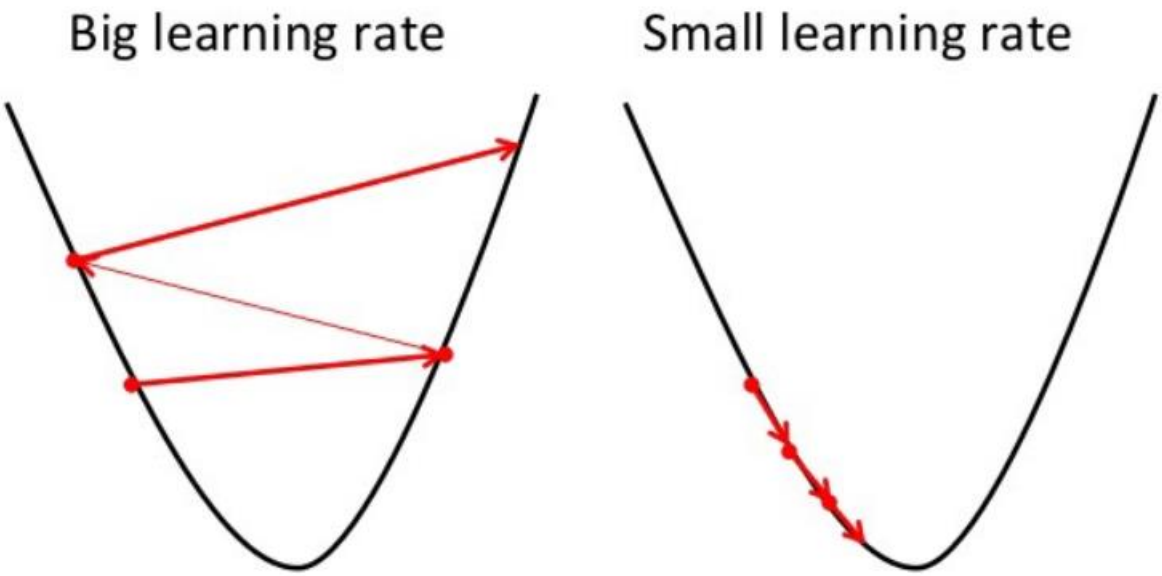
2. **Number of hidden units per layer**:-It was also selected reasonably to find a right spot between high bias and variance. It also depends on the data size used for training. Hidden units in our models were in general powers of 2 between 128 to 1024 and were used in different combinations with different models.

3. **Activation Function**:- Our choices in this are ReLU, Sigmoid & Tanh.



4. **Optimizer**:- It is the algorithm used by the model to update weights of every layer after every iteration. Popular choices are SGD,RMSProp and Adam. All three have different properties and are used according to the data . We achieved great results by using RMSProp and Adam.

5. **Learning Rate**:- It is responsible for the core learning characteristic and we chose it in such a way that it is not too high wherein the model is unable to converge to minima and not too low such that the model is unable to speed up the learning process. We tried in powers of 10, specifically 0.001,0.01, 0.1,1.



6. **Batch Size**:- It is indicative of the number of patterns shown to the network before the weight matrix is updated. If batch size is less, patterns would be

less repeating and hence the weights would be all over the place and convergence would become difficult. If batch size is high learning would become slow as only after many iterations will the batch size change. We tried batch sizes in hta range of 32 to 512 and in general the batches which were power of 2 gave the best results.

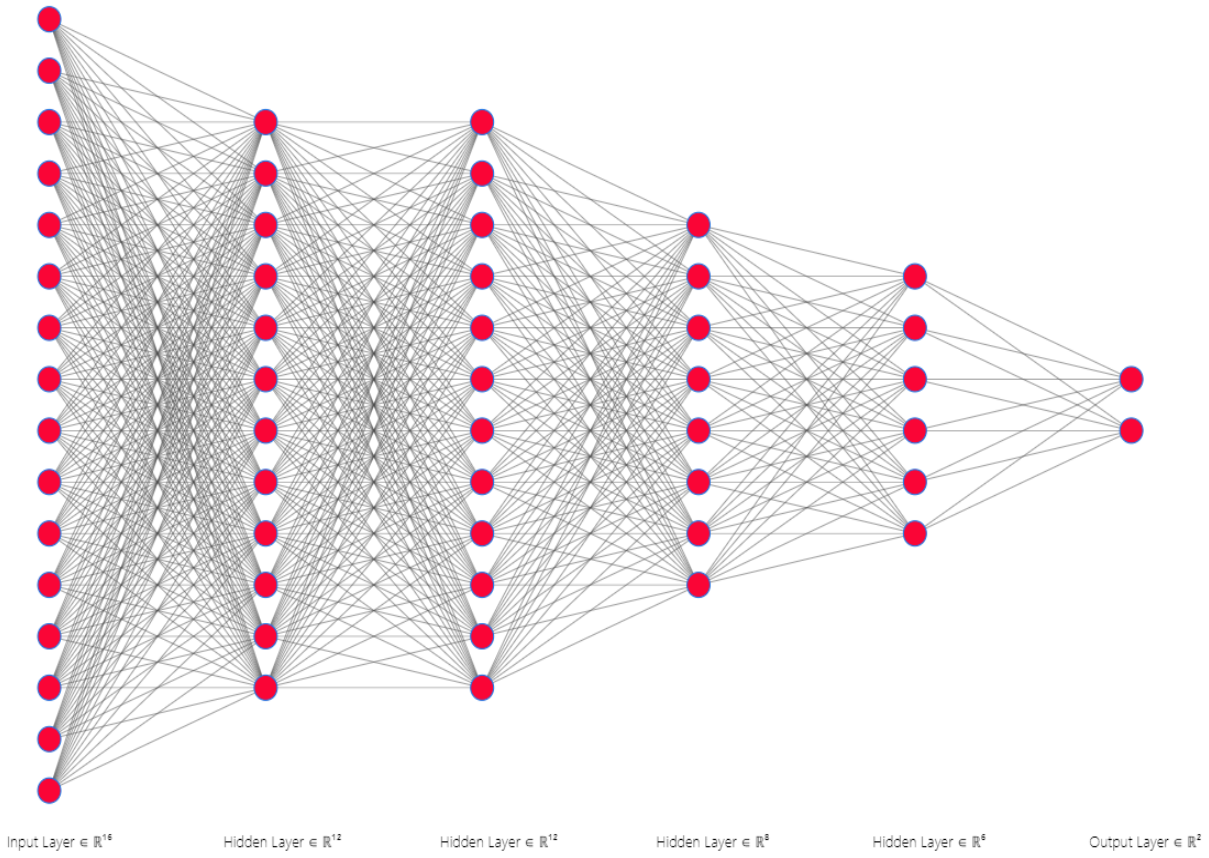
7. **Number of Epochs**:- The number of epochs is the number of times the entire training data is shown to the model. It played an important role in how well the model fits on the train data. High number of epochs overfitted on our data .Lower number of epochs also limited the potential of the model leading to underfitting. A large number of different epochs were used while training the models and many lead to overfit or underfit the data . Epochs in the range of 18 to 23 gave the best results.

8. **Dropout**:- The keep-probability of the Dropout layer can be thought of as a hyper-parameter which could act as a regularizer to help us find the optimum bias-variance spot. While using dropout the models drop certain connections every iteration therefore the hidden units cannot depend a lot on any particular feature. The values it can take can be anywhere between 0 to 1 and it is solely based on how much the model is overfitting. When we used a 5 layer deep neural network there was a huge problem of overfitting which we tried to overcome using high dropouts . In general while using 2 layer deep neural networks we experimented with dropout's keep probability values between 0.2 to 0.4.

9. **L1/L2 Regularization**:- It serves as another regularizer wherein the very high weight values are curbed so that the model is not dependent on a single feature. This generally reduces variance with a trade-off of increasing bias i.e. lowering accuracy. We experimented with both types of regularizations techniques and after feature selection we generally used L2 regularization.

Visual Representation of models

1. Overfitted model

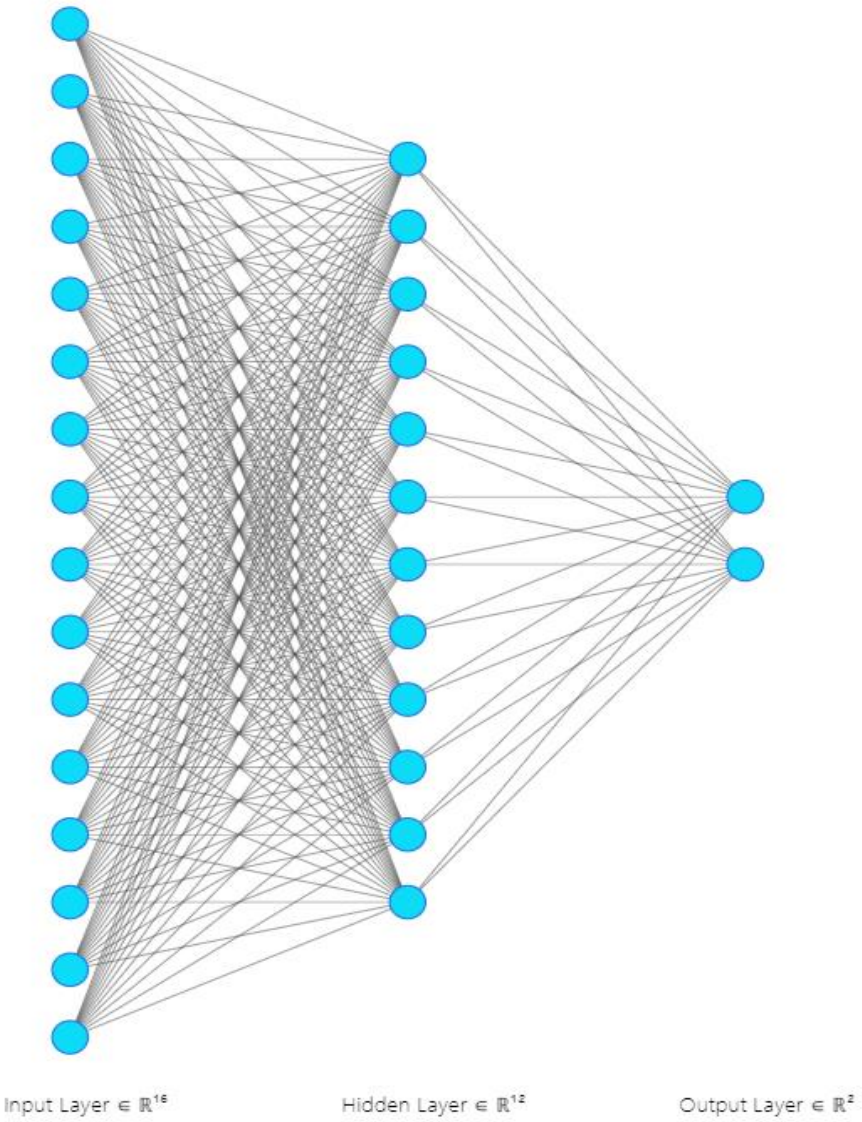


Number of hidden layers:-4

No. of nodes in each layer :- 512 , 512 , 300 , 256

Activation :- Relu , Optimizer :- Adam , Learning rate :-0.003

2. **Under Fitted model**

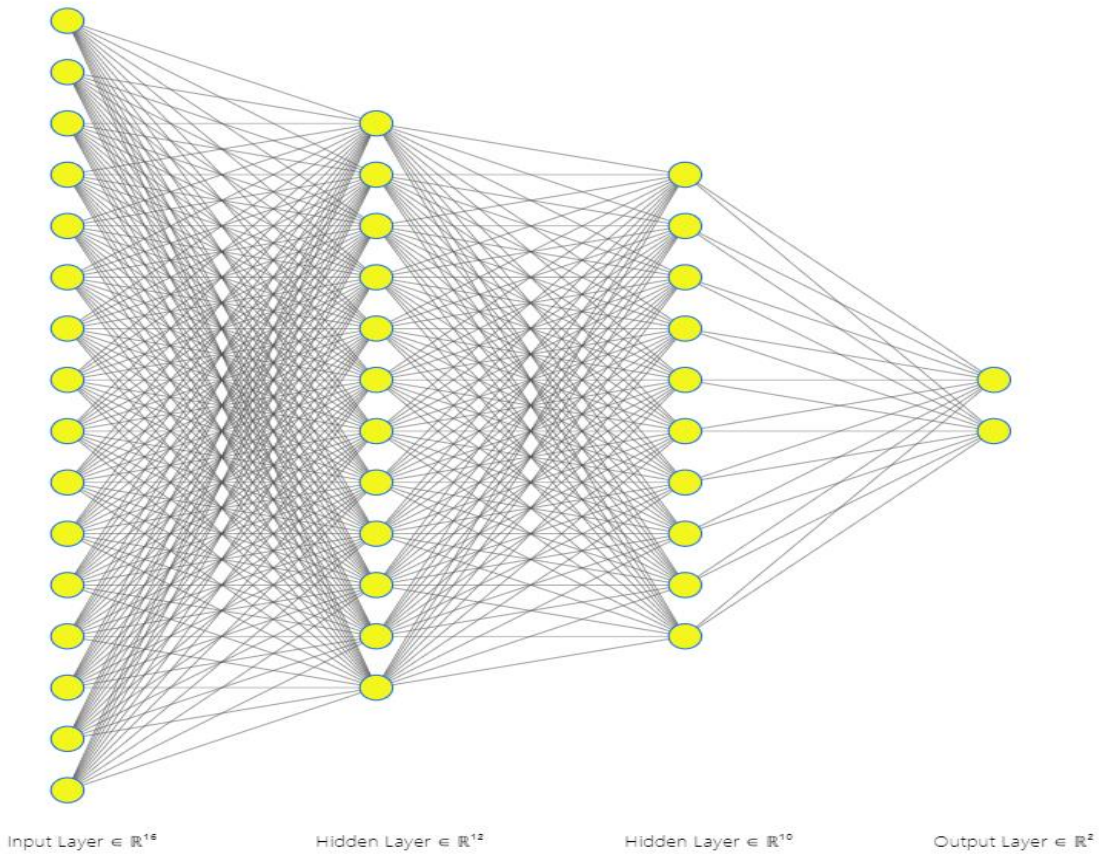


Number of hidden layers:-1

No. of nodes in each layer :- 512

Activation :- Tanh , Optimizer :- RMSProp , Learning rate :-0.2

3.Best Fit model



Number of hidden layers:-2

No. of nodes in each layer :- 512 , 512

Activation :- Relu , Optimizer :- Adam , Learning rate :-0.003

Dropout of 0.3 in both hidden layers.

Course Recommendation System

Four types of recommendation systems have been implemented which are described as follows-

1. Simple Recommender -

The Simple Recommender offers generalized recommendations to every user based on course popularity. The basic idea behind this recommender is that courses that are more popular and more critically acclaimed will have a higher probability of being liked by the average users. The main drawback of this recommender system is that it does not give personalized recommendations based on the user.
In a simple recommendation system, weighted ratings were calculated and on the basis of that recommendations were made.

$$\text{Weighted Rating (WR)} = (v/(v+m))*R + (m/(v+m))*C$$

where,

- v is the number of votes for the course
- m is the minimum votes required to be listed in the chart
- R is the average rating of the course
- C is the mean vote across the whole report.

2. Content Based Recommender -

To personalise our recommendations more, an engine that computes similarity between courses based on certain metrics has been built and courses that are most similar to a particular course that a user liked were suggested.
Two content based recommender system were implemented based on certain metrics-

1. Tf idf vectorizer was applied on the description of the courses and similarity scores were retrieved by applying cosine similarity on the tf idf matrix.
2. Tf idf vectorizer was applied on course organization, course instructor and similarity scores were calculated by cosine similarity.

User's taste and history were not considered in this recommender system.

3. Collaborative filtering -

Collaborative Filtering is based on the idea that users similar to us can be used to predict how much we will like a particular product or service those users have used/experienced but we have not. Singular value decomposition (SVD) was used to predict how much rating a user will give to a particular course.

4. Hybrid Recommender -

In this system, all three recommender systems were combined to obtain optimised course recommandations.
First, recommendations from content based recommenders were retrieved and then sorted retrieved courses on the basis of weighted ratings obtained from a simple recommender system and then predicted the ratings of these retrieved courses that a particular user will give using collaborative filtering. In this way, this system considers the user's taste, user’s history and popularity of courses.

Algorithms

Tf-idf vectorizer -

TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is a very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithms for prediction.

Term Frequency -
The number of times a word appears in a document divided by the total number of words in the document. Every document has its own term frequency.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Inverse Data Frequency (IDF) -

The log of the number of documents divided by the number of documents that contain the word w. Inverse data frequency determines the weight of rare words across all documents in the corpus.

$$idf(w) = log(\frac{N}{df_t})$$

TF-IDF is simply the TF multiplied by IDF.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

1. **Cosine similarity -**

   Cosine Similarity is a measurement that quantifies the similarity between two or more vectors.The cosine similarity is the cosine of the angle between vectors.

   The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Results

Results from simple recommender system -

In this system, an input field is given and on the basis of weighted ratings courses are recommended.

Let's say input field is "Statistics" then our system recommends following courses-

Courses	Course Organisation	Weighted Ratings
Business Statistics and Analysis	Rice University	8.21
Introduction to Statistics & Data Analysis in Public Health	Imperial College London	8.1
Methods and Statistics in Social Sciences	University of Amsterdam	7.98
Statistics with Python	University of Michigan	7.95
Statistics with R	Duke University	7.93

Results from content based recommender system -

In this system, an input course is given and similar courses with respect to input are recommended on the basis of similarity metrics.
Let's say input course is "Introduction to Data Science in python by university of michigan" then our system recommends courses that are shown in the table below:

Courses	Organization
Advanced Data Science with IBM	IBM
Data Science	Johns Hopkins University
Mathematics for Data Science	National Research University Higher School of Economics
Applied Machine Learning in Python	University of Michigan
Databases and SQL for Data Science	IBM

Results from collaborative filtering -

In collaborative filtering , course rating is predicted by using the SVD algorithm. For a particular user, ratings of all the courses for which the user has not given rating were predicted and sorted in descending order of these predicted ratings to recommend courses. This model tries to predict ratings based on how the other users have predicted the course. Two inputs are taken by this system from the dataset-

1. Course_id
2. User_id

In the code snippet shown below, the function takes predictions from SVD and how many courses to recommend as inputs and shows top recommendations to each user.

```
from collections import defaultdict
def get_top_n(predictions, n):
  top_n = defaultdict(list)
  for uid, iid, _, est, _ in prediction:
    top_n[uid].append((iid, est))

  for uid, user_ratings in top_n.items():
    user_ratings.sort(key = lambda x: x[1], reverse = True)
    top_n[uid] = user_ratings[:n]
  return top_n
  pass
top_n = get_top_n(prediction, n = 3)
for uid, user_ratings in top_n.items():
  print(uid, [iid for (iid, rating) in user_ratings])
```

```
1 [1272, 898, 904]
2 [1272, 1197, 1252]
3 [356, 2571, 134853]
4 [1272, 26776, 3347]
5 [858, 162, 1272]
6 [34405, 68954, 2959]
7 [745, 1212, 1148]
8 [750, 260, 1272]
9 [858, 4973, 7153]
10 [1278, 954, 1222]
```

Results from hybrid recommender -

This recommender system takes two inputs - User_id and course.

Let's say two inputs are - (1, "Supply Chain Finance and Blockchain Technology") then recommendations for user with user_id 1 are shown below-

Courses	Estimated Rating
Foundational Finance for Strategic Decision Making	8.4
Finance for Non-Finance Professionals	8.10
Corporate Finance Essentials	8.05
Behavioral Finance	8.03
Understanding Modern Finance	7.98

CONCLUSION

In our overall BTP work, a detailed & comprehensive analysis of data form e-learning platform was performed. Analysis of the data showed how a lot of inferences can be drawn from the raw data. Raw data cleaning was done which included working on missing value, encoding categorical variables and feature scaling of the variables.

For feature extraction PCA and RFE(Recursive Feature Elimination) was used which greatly affected the models performance. The extracted features were then used as input variables for 5 classification models which contained a huge range of hyper parameters. Tuning of those parameters revealed the facts of how greatly a model is affected by these hyper parameters. The SVM model using the linear kernel resulted in the highest accuracy of 92.37%. But as it turns out the other models like Logistic Regression (accuracy of 91.36%), K Nearest Neighbour (accuracy of 91.06%) and Random Forest (accuracy of 91.11%) all showed an accuracy very close to SVM. The results showed that even though there are a great number of different classification algorithms which are all great at making predictions, one of the major factors which greatly influenced the accuracy is how data preprocessing is done. Features extraction and feature selection plays a vital role in models performance.

Classification of customers was also done using Neural Networks which contained a lot of hyperparameters . With the correct sets of hyperparameters like the number of hidden layers, nodes in each layer, the learning rate , type of optimization and activation functions etc played a vital role in the performance of the model. Deep neural networks tend to overfit on the training data which we overcame by using dropouts and different regularization techniques.

Using the data of different courses on the website along with user specific data we also developed a recommendation system which can suggest the most likely courses the user will be interested in . This recommended course data can be helpful to the sales team which can further increase the conversion rate as these courses will be tailored towards the user that already has a high probability of buying the course. Hence with this the targeted user specific system the courses bought per user can also be increased along with the conversion rate.