**University admit eligibility predictor**

**PROBLEM SOLUTION FIT**

**Problem Statement: University admit eligibility predictor**

Every year thousands of college graduates apply for the universities from all around the world.  Applying to foreign universities is not an easy task, it involves many steps and procedures to follow.  Choosing the right universities or colleges is definitely an another hurdle students have to face. Many students apply for the universities in which they have little chance of acceptance. What if there is a system that could guide students and recommend best universities list and predict their admission chance in those universities according to their profile and scores. So, the idea behind ‘University Admit Eligibility Predictor system’.

**SOLUTION:**

These problems can be resolved by using regression algorithms / classification algorithms as they can consider most of the features for prediction. Linear regression / KNN classification / Random forest Regressor can be used as the machine learning model for the model. XG boost model can also be used which performs better on small to medium scale datasets but the model giving accurate and desired results only will be selected. The aim of the proposed system is to address the limitations of the current system. The requirements for the system have been gathered from the defects recorded in the past and also based on the feedback from users of previous metrics tools.

Following are the objectives of the proposed system:

• Reach to geographically scattered student.

• Reducing time in activities

• Paperless admission with reduced man power

• Operational efficiency

**PROCEDURE:**

Step 1: Importing Necessary Modules/Libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from keras.models import Sequential

from keras.layers import Dense ,Dropout,BatchNormalization

from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasRegressor

Step 2: Loading Dataset Into the Program

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| df = pd.read\_csv('Admission\_Predict.csv')  df.head()  Step 3: Data Pre-processing and Data Splitting  Before building our main model, we would require some pre-processing which involves dropping any column  which is not necessary for the model.  Here the column ‘Serial No.’ isn’t necessary for admission prediction so we drop it out of the data.   |  | | --- | | df**=**df.drop("Serial No.",axis**=**1) |   After this we would be dividing the dataset into X and Y sub-datasets where X will have all the information  and Y will include the final probability.   |  |  | | --- | --- | |  | Y**=**np.array(df[df.columns[**-**1]])  X**=**np.array(df.drop(df.columns[**-**1],axis**=**1)) |   Now, the next step is to split the dataset into training and testing datasets using the 80:20 train test split rule  where 80% of the data is used for training and the rest 20% is used for testing.   |  | | --- | | X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,Y, test\_size**=**0.2, random\_state**=**0) | |

Step 4: Training of the Model

The next step is to create out the model object and train the same on the training dataset as mentioned in the code below. You can keep the number of epochs according to your own preference.

**Approach:**

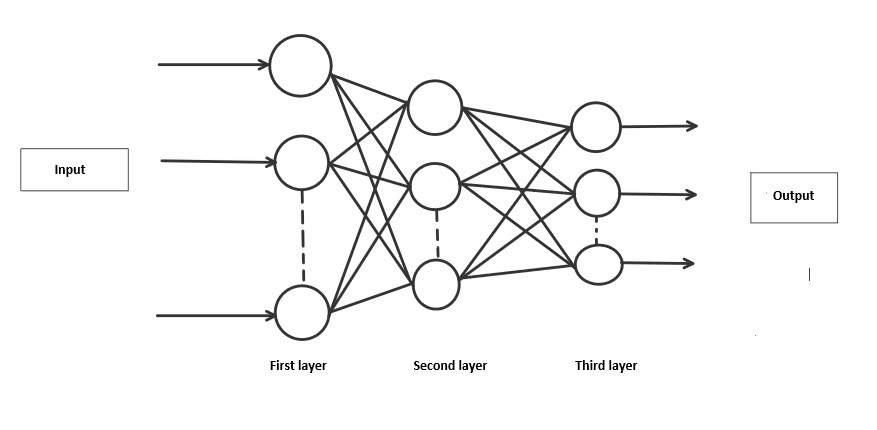
We will approach this project by using a three-layered Neural Network.

* **The input layer:** It distributes the features of our examples to the next layer for calculation of activations of the next layer.
* **The hidden layer:** They are made of hidden units called activations providing nonlinear ties for the network. A number of hidden layers can vary according to our requirements.
* **The output layer:** The nodes here are called output units. It provides us with the final prediction of the Neural Network on the basis of which final predictions can be made.

A neural network is a model inspired by how the brain works. It consists of multiple layers having many activations, this activation resembles neurons of our brain. A neural network tries to learn a set of parameters in a set of data which could help to recognize the underlying relationships. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

**METHODOLOGY:**

We have implemented a Neural Network with 1 hidden layer having 100 activation units (excluding bias units). The data is loaded from a .mat file, features(X) and labels(y) were extracted. Then features are divided by 255 to rescale them into a range of [0,1] to avoid overflow during computation. Data is split up into 60,000 training and 10,000 testing examples. Feedforward is performed with the training set for calculating the hypothesis and then backpropagation is done in order to reduce the error between the layers. The regularization parameter lambda is set to 0.1 to address the problem of overfitting. Optimizer is run for 70 iterations to find the best fit model.



**ALGORITHM:**

**Forward Propagation Architecture:**

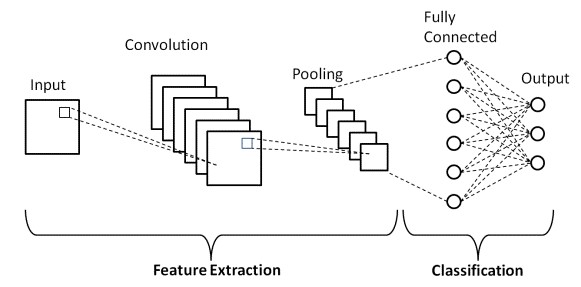
It is a small workflow of how CNN module will extract the features and classify the image based on it. The architecture shows the input layer, hidden layers and output layer of the network. There are many layers involved in the feature extraction phase of the network which involves convolution and subsampling .

**EXPLANATION OF THE PROPOSED SYSTEM**

* The first layer of the architecture is the User layer. User layer will comprise of the people who interacts with the app and for the required results.
* The next three layers is the frontend architecture of the application.

The application will be developed using which is the open-source platform for HTML, CSS and JavaScript.

The application is deployed in the localhost which is shown on the browser. Through the app, the user will be able to upload pictures of the handwritten digits and convert it into the digitalized form. • The one in between the database and view layer is the business layer which is the logical calculations on the basis of the request from the client side. It also has the service interface. • The backend layer consists of two datasets: Training Data and Test Data. The MNIST database has been used for that which is already divided into training set of 60,000 examples and test of 10,000 examples. • The training algorithm used is Convolution Neural Network. This will prepare the trained model which will be used to classify the digits present in the test data. Thus, we can classify the digits present in the images as: Class 0,1,2,3,4,5,6,7,8,9.



**WORKING**

* Neural Networks receive an input and transform it through a series of hidden layers.
* Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer.
* Neurons in a single layer function completely independently. • The last fully connected layer is called the "output layer“.

**Convolution Layer**: The Convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume.

During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2- dimensional activation map of that filter.

As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input..

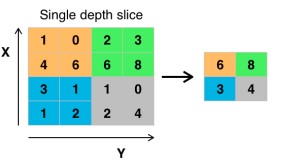
**Feature Extraction**:

All neurons in a feature share the same weights .In this way all neurons detect the same feature at different positions in the input image. Reduce the number of free parameters.

**Subsampling Layer:** Subsampling, or down sampling, refers to reducing the overall size of a signal .The subsampling layers reduce the spatial resolution of each feature map. Reduce the effect of noises and shift or distortion invariance is achieved.

**Pooling layer:** It is common to periodically insert a Pooling layer in-between successive Conv layer in a Convent architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.

**TensorFlow**: TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. See the sections below to get started. By scanning the numerical digit and convert into png format using python3 command in terminal we can get text output and sound output.



**Pooling layer**

**RESULT:**

As with any work or project taken up in the field of machine learning and image processing, we are not considering our results to be perfect.

Machine learning is a constantly evolving field and there is always room for improvement in your methodology; there is always going to be another new approach that gives better results for the same problem. The application has been tested using three models: Multi-Layer Perceptron (MLP), Convolution Neural Network (CNN). With each model we get a different accuracy of the classifier which shows which one is better.

**SOLUTION ARCHITECTURE**

