**Task 1: Understanding and analysing various machine learning models**

**SubTask 1:**

The main goal for doing the weather prediction is to predict weather it will rain or not. There are various factors that inflences the weather, some of the factors/features include temperature, Wind Speed, Pressure, Humidity and Wind Direction. The weather prediction is often classified as a binary classification problem. Hence, we can say that it has only two labels, in our case the two labels are Raining and not Raining. Let us look at some static classification methods that can help us to predict the rainfall.

**The static classification methods:**

**Naïve Bayes Classifier (BN):**

It is a probabilistic classifier based on Bayes theorem. The basic concept is that it relates the probability of a class given some observed features to the conditional probability of those features given the class and the prior probability of the class.

**Working**:

The important thing in naïve bayes is, each of the features from the dataste are assumed to be independent given the class The weather prediction has different features, these features are denoted as (X) and we have to predict the probability of a certain class, such as Rain which is represented as

The likelihood is . Each of these are calculated using the Guassian distribution (Fig. 1). is the probability of being rain, this is calculated using the given data. The Normalizing constant/evidence is calculated by summing over all possible features available associated with the class.

A math equations on a white background

Description automatically generated

Fig. 1

The model is built be importing the NaïveBayesClassifier from the scikit-learn library. The data is split, and the model is trained using the train dataset.

**k-nearest neighbour(KNN):**

The KNN is an instance-based learning that makes predictions based on the majority class for classification that are grouped together among its k- nearest neighbours in the feature space. KNN uses a distance metric, to determine the proximity between data points(closeness). The distance metric differs for each application, typically Euclidean distance or Manhattan distance is used.

**Working:**

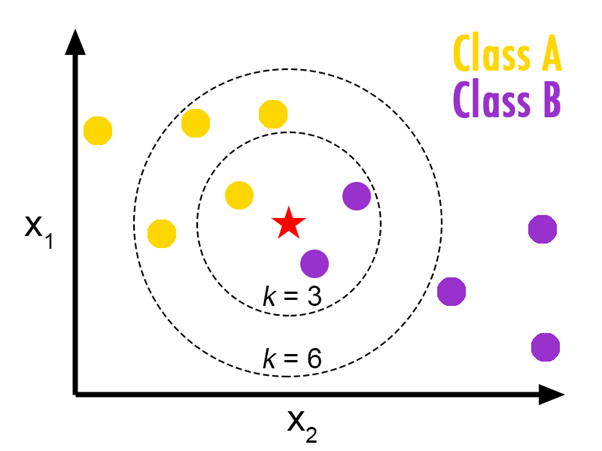
In the context of weather prediction, we can take two features which mostly influences the rain. The distance between those features is calculated using the Euclidean distance.

A math equations and formulas

Description automatically generated with medium confidence

After calculating the distance, the new data points are assigned a class and we must sort them in order and pick the class nearest to the specified k value to make the prediction. K is a hyperparameter that we must determine and it can significantly impact the model's performance. From the image, if k = 3, and if the majority of the neighbours are “rain” (Class A), the prediction is “rain” but if most are “no rain” (Class B), then the prediction is "no rain”.

The model is built be importing the KNeighborsClassifier from the scikit-learn library. The data is split and the model is trained using the train dataset.



**Ensemble Learning(EL):**

There are many types of Ensemble learning methods available, but for weather prediction, we have chosen Random Forest as it produces accurate predictions for classifications. The base model of the random tree is Decision tree, it compiles many trees and combines the result to generate the output by voting.

**Working:**

In context to the weather prediction, we have a dataset D consisting of N points with features X and labels y.

.

We now have to create multiple bootstrap samples D1, D2,..,Dk from the training dataset. Now we create several decision trees on the bootstrap training samples. A Random subset of features is considered for splitting. Spilt the node into two children nodes. Repeat the same process for splitting. This ensures that not all features are used at every decision point. Now we have to combine the outputs for prediction, For classification, each tree casts a vote for the class label. The class with the most votes is the final prediction.

Random Forest provides a feature importance score based on how much each feature contributes to the reduction in impurity (e.g., Gini impurity) when making splits in the trees.

**Support Vector Machines(SVM):**

SVM can performs linear and nonlinear classification. It aims to find the hyperplane that separated the data into different classes using the training instances while maximizing the margin between the classes. The margin is the perpendicular distance from the hyperplane. The Support vectors are the data points closest to the hyperplane.

**Working:**

In contest to the weather prediction, we need to find the hyperplane that separates the classes, it is represented as,

A black text on a white background

Description automatically generated

The SVM model requires to define the optimization problem,

A group of math equations

Description automatically generated

The SVM also uses a kernel function that helps to calculate the inner products, they help to capture the relationship between features. The standard kernels that are used are linear, polynomial, radial basis function and sigmoid.

A math equations on a white background

Description automatically generated

The RBF kernel is suitable for the weather prediction as it can handle non-linear relationships, it transforms the feature space to a higher dimensional space. Then we calculate the optimization problem equation to find the values of the and b to draws the hyperplane. Now calculate the decision function, . If its greater than zero, then its raining otherwise its not raining.

The model is built be importing the SVM() classifier from the scikit-learn library. The data is split and the model is trained using the train dataset.

**SubTask 2:**

**cGAN:**

The cGAn introduces conditional information during the training and generation processes. The condition provides a way to control and guide the generation process. The condition can be a class label, text description or any other relevant information. The generator G in cGAN captures the data distribution, it takes the noise to data space and also y which is the condition in hidden layer as input. The output generated data will a conditioned relevant data. The discriminator in a cGAN t estimates the probability of a sample that came from the training data.



**wGAN:**

wGAN provides stability to learning and gets ride of problems such as mode collapse and instability. It introduces the Wasserstein distance. It introduces new loss function called the Wasserstein loss, its main goal is to reduce the distance that produce that is progressively closer to the true data distribution. To ensure the proper functionality of the Wasserstein distance, the Lipschitz constraint is introduced, we enforce weight clipping on the discriminator. This keeps the discriminator's weights within a certain range.

**Task 2: Develop learning-based model(s) for sequence classification**

**Subtask 1:**

**Introduction:**

To develop a learning-based model to classify the sequential data provided in the Human Recognition Activity dataset, a clear understand about that dataset is essential. The Human Recognition Activity dataset contains data about a common person’s activities, these data are sourced using the sensors in the smartphones. Common activities in the dataset includes Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing and Laying.

Smartphones are used to gather the sensor data, such as body acceleration values and gyroscope readings from accelerometers and gyroscope sensors. Each row of data is taken from many different individuals. This dataset most popular and is widely being used in the field of machine learning and activity recognition.

For our activity prediction, we have made use of the CNN, LSTM, and bi-direction recurrent network (Bi-RNN) models. We will discuss about these models individually later on. First in order to build our models, we need to import the required python libraries. As we are working with a dataset, we need to import Pandas and NumPy Libraries. If we are going to implement graphs, then we need to import matplotlib. In the context of building a model, we need to import TensorFlow to access the necessary functions and classes of each Models.

**Load Data:**

Next we have to load the data, the dataset is given in a zip folder. The train data is being stored in a separate folder and the test data is stored in a separate folder. The X\_train contains the accelerometer values and gyroscope values. Each sensor values are given in three separate text files, the reason is that these are typically measured along the X, Y, and Z axes, representing different spatial directions. To load these data, we need to stack the separate text files into one dataset. The X\_test also follows the same procedure. The activity labels are contained in the Y\_train and Y\_test text files. We simply just load these data directly to their corresponding variables by provided the file path.

**Data preprocessing:**

Data preprocessing is an important step in machine learning as it greatly influences the model’s performance and prediction. As a first step in data preprocessing, we have to check for any missing values in the dataset. If there are no missing values, then we can proceed with the next step if not then either we have to fill the column with a relevant value or drop those rows entirely. In our case, we do not have any missing values. The next part of the process would be Feature Scaling. The data that we have does not contain features with varying scale nor we have to ensure all values fall within a specific range. Hence, we have avoided doing Feature Scaling. Next we are doing One-Hot encoding, this is categorized as a type of data preprocessing. This is done when we have more than one class when doing a classification. It represents the labels as one hot vectors, which in turn treats them as distinct, categorical classes which is helpful when training the model. Now we can use the dataset to train models.

**CNN Model Architecture:**

The Convolutional Neural Network is most commonly used for image classification. The reason why it is being used here is because they are capable of extracting features from the provided dataset using the convolution and pooling layers. CNN is proven to be effective in analysing time series data and classifying activity. The architecture consists of 3 convolution layers followed by the 2 MaxPool layers and then 2 dense layers.

A screenshot of a computer code

Description automatically generated

The first two convolution layers contain 32 and 64 filters respectively with a kernel size of 3×3 and as an activation function, we have used ReLU. The MaxPool layer were given a pool size of 2 as parameter. In between these layers we have introduced a dropout layer with a value of 0.5, the main function of this layer is to prevent overfitting. Finally, we used a dense layer with 256 & 128 neurons respectively, it also uses ReLU as its activation function which is followed by a final output Dense layer with n\_output, which are the features and SoftMax as an activation function.

After building the model, we compile the model and train the model with the X\_train and Y\_train dataset to produce the accuracy result for that model using the test dataset.

**LSTM Model Architecture:**

Long Short-Term Memory is a type of improved version of the recurrent neural network. The LSTM model is well suited for training sequential data. They can capture certain features or order dependencies within the data.

A close-up of a computer code

Description automatically generated

In the LSTM Architecture, we made use of a single LSTM layer. The LSTM layer contains 64 neurons and the input is also given as an argument. Next we have introduced a Dropout layer with a value of 0.5 to prevent overfitting, it specifies that 50% of the neurons will be randomly dropped from the previous layer and then the remaining are inputted into the next layer. The next layers are made up of Dense layers which contains 64 neurons and ReLU is given as its activation function. The activation function for the final layer is Softmax, it is commonly given for multi-class classification tasks.

After building the model, we compile the model and train the model with the X\_train and Y\_train dataset to produce the accuracy result for that model using the test dataset.

**Bi-RNN Model Architecture:**

In the Bi-RNN Architecture, we have made use of a Bideirectional LSTM layer. The Bideirectional LSTM processes the input in both forward and backward directions. It contains about 64 neurons. The Next layer is the dropout layer which has a value of 0.5 to prevent overfitting. The next layer contains the dense layer with 100 neurons and ReLU as its activation function. The next layer is the final layer made with n\_outputs as its neurons and Softmax as its activation function.

A screenshot of a computer program

Description automatically generated

After building the model, we compile the model and train the model with the X\_train and Y\_train dataset to produce the accuracy result for that model using the test dataset.

**SubTask 2:**

Let us have a look at the accuracy and loss values for each model,

|  |  |  |
| --- | --- | --- |
| Models | Accuracy | Loss |
| CNN | 85% | 0.962 |
| LSTM | 84% | 0.541 |
| Bi-RNN | 86% | 0.465 |

From the above table, we can easily infer that Bi-RNN has the best accuracy and the lowest loss value. The accuracy and the loss of each model can be even greatly increased if we insert more layers in each of the model and also include some generalization techniques to prevent them from overfitting.

Sometimes, if the given dataset is not sufficient then our models may perform poorly, but in our case we have ample amount of data sample. When it comes to prediction of activities using the test data, we can say that the our models performs pretty good. In order to evaluate our model, I have implemented confusion matrix and also included the classification report for each of the models, they will help us to infer how successful the models has predicted the activity classes. The confusion matrix of each models are as follows,

**Confusion Matrix of CNN:**

A diagram of a graph

Description automatically generated

**Confusion Matrix of LSTM:**

**A colorful squares with white text

Description automatically generated**

**Confusion Matrix of Bi-RNN:**

A chart with numbers and symbols

Description automatically generated with medium confidence

**Classification Report:**

The Classification report will help us to how precise the predictions are made for the test dataset. For instance, let us take the classification report of CNN. We can see that the Walking activity has a precision of 99%, which means that the CNN model can accurately predict that activity.

A screenshot of a computer

Description automatically generated

**Reference:**

1. https://medium.com/geekculture/human-activity-recognition-with-smartphones-using-deep-learning-73a7867b2e23#id\_token=eyJhbGciOiJSUzI1NiIsImtpZCI6ImEwNmFmMGI2OGEyMTE5ZDY5MmNhYzRhYmY0MTVmZjM3ODgxMzZmNjUiLCJ0eXAiOiJKV1QifQ..r1uTpgkTVv8V\_2WqVNjTnkSMPxRyLeHBtWOxrLULRfVHxrE7b1LWImdZieDwhAywcRRbnW5SH7EglaK-aBT0qlX1SCzRXQpqrzyPd9SsvjWM8Sq41t5ddDmPmQzlQK1pnoqRWYmP73SUGjFzgqv7V4l2c1wsEB5GgwPgUyHSfmIN-2rWX5EzL\_6XgKZfj2sdW8Z7UB8XNJaqp8smjFhs9OvO3NjaB5RKEHrgvyrCWrGOAGiTxDEchgkVW7aL987u24fz5FNVuxAWXsV9N2gLJFdoVDXCrehy3ggFYwfXbxz4vJwmLr\_r0tqjuoJNDb-CqpqlSFNvK4UkRb4--Vm1Ow
2. https://machinelearningmastery.com/how-to-model-human-activity-from-smartphone-data/