## CHAPTER 1

### About the company - IBM

* 1. **About IBM**

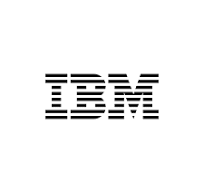
**International Business Machines Corporation (IBM)**, commonly known as IBM, is a multinational technology and consulting company headquartered in Armonk, New York. Founded in 1911 as the Computing-Tabulating-Recording Company (CTR), it was renamed International Business Machines in 1924. Over its long history, IBM has evolved from a hardware and software provider to a major player in the fields of cloud computing, artificial intelligence (AI), and consulting services.

* 1. **Key Areas of Operation**

**Cloud Computing** IBM Cloud offers a comprehensive suite of cloud computing services, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IBM's hybrid cloud approach allows businesses to integrate their public and private cloud environments seamlessly.

**Artificial Intelligence** IBM is a leader in AI, largely through its IBM Watson platform, which provides AI solutions for various industries. Watson's capabilities include natural language processing, machine learning, and data analytics, helping organizations make data-driven decisions.

**Consulting and IT Services** IBM Global Services provides IT consulting, business process outsourcing, and systems integration services. These services help clients optimize their IT infrastructure, improve operational efficiency, and drive digital transformation.



**Fig 1.1: IBM Logo**

## CHAPTER 2

### Introduction to domain of internship

* 1. **Data Science**

**Data Science** is a multidisciplinary field that involves extracting insights from data using various techniques from statistics, computer science, and domain knowledge. The key steps typically involved in a data science workflow are data collection, data cleaning, data analysis, and data visualization.

**Data Loading and Cleaning**:

df = pd.read\_csv("L A\_Weather.csv")

print(df.shape)

**Data Exploration and Visualization**:

fig = plt.figure(1)

ax = fig.add\_subplot(111, projection="3d")

ax.scatter(Tmax, Tmin, Tavg, marker='o')

ax.set\_xlabel('Max Temp')

ax.set\_ylabel('Min Temp')

ax.set\_zlabel('Average Temp')

plt.show(block=False)

* 1. **Machine Learning**

**Machine Learning** involves building models that can learn from data and make predictions. This domain encompasses supervised learning, unsupervised learning, and reinforcement learning.

**Data Preprocessing**:

scaler = MinMaxScaler()

scaler.fit(Temp)

Temp = scaler.transform(Temp)

scaler1 = MinMaxScaler()

scaler1.fit(Tavg)

Tavg = scaler1.transform(Tavg)

**Splitting Data**:

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(Temp, Tavg, test\_size=0.3)

**Building and Training the Model**:

model = Sequential() model.add(Dense(32, activation='relu', input\_dim=2)) model.add(Dense(32, activation='relu')) model.add(Dense(1, activation='sigmoid')) model.compile(loss='mean\_squared\_error', optimizer='rmsprop', metrics=[metrics.mean\_absolute\_error]) model.fit(X\_train, Y\_train, epochs=500, batch\_size=32, verbose=2)

## CHAPTER 3

### Technology and Tools

#### 3.1 ****Python Libraries****

* **os**
  + The os module in Python provides functions for interacting with the operating system. In this script, it is used to set an environment variable to disable oneDNN optimizations in TensorFlow.
* **numpy**
  + NumPy is a fundamental package for scientific computing in Python. It provides support for large multidimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Here, NumPy is used for array manipulation and data processing.
* **pandas**
  + Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames to handle and process structured data. In this script, Pandas is used to read and process the CSV file containing the Los Angeles weather data.
* **matplotlib**
  + Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. Here, it is used to create 3D scatter plots and 2D plots to visualize the weather data and the model predictions.
* **mpl\_toolkits.mplot3d**
  + This is a toolkit for Matplotlib to create 3D plots. In this script, it is used to create a 3D scatter plot of the weather data.

#### 3.2 ****Scikit-learn (sklearn)****

* **preprocessing.MinMaxScaler**
  + MinMaxScaler is a preprocessing tool that scales the features to a given range, typically between 0 and 1. This scaling is crucial for neural networks to ensure that the input features are on a similar scale. Here, it is used to scale the temperature data.
* **model\_selection.train\_test\_split**
  + This function splits the dataset into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate the model's performance. Here, it is used to split the temperature data and corresponding average temperatures into training and testing datasets.

#### 3.3 ****Keras with TensorFlow Backend****

* **Sequential Model**
  + Keras provides a high-level neural networks API. The Sequential model is a linear stack of layers, making it easy to build and train neural networks. Here, it is used to build the weather prediction model.
* **Dense Layers**
  + Dense layers are fully connected layers where each neuron receives input from all neurons of the previous layer. In this script, three dense layers are used:
    - The first dense layer with 32 neurons and ReLU activation function.
    - The second dense layer with 32 neurons and ReLU activation function.
    - The output layer with 1 neuron and sigmoid activation function.
* **metrics**
  + Keras metrics are used to evaluate the performance of the model. Here, mean\_absolute\_error is used to monitor the training process.
* **Model Compilation and Training**
  + The model is compiled with mean\_squared\_error as the loss function and rmsprop as the optimizer. The model is then trained on the training data for 500 epochs with a batch size of 32.

#### 3.4 ****Weather Data Analysis and Visualization****

* **Loading Data**
  + The weather data for Los Angeles is loaded from a CSV file using Pandas. The data includes average temperature (Tavg), maximum temperature (Tmax), and minimum temperature (Tmin).
* **Data Visualization**
  + A 3D scatter plot is created to visualize the relationship between Tmax, Tmin, and Tavg. This helps in understanding the data distribution and potential patterns.
* **Data Preprocessing**
  + The temperature data (Tmax and Tmin) is concatenated and transposed to prepare it for scaling. MinMaxScaler is used to scale both the input features (Tmax and Tmin) and the target variable (Tavg).
* **Model Training and Prediction**
  + The preprocessed data is split into training and testing sets. The neural network model is trained on the training set and then used to predict Tavg on the test set.
* **Result Visualization**
  + A scatter plot is created to visualize the predicted Tavg versus the actual Tavg from the test set. Additionally, a line plot is used to compare the predicted and real data.

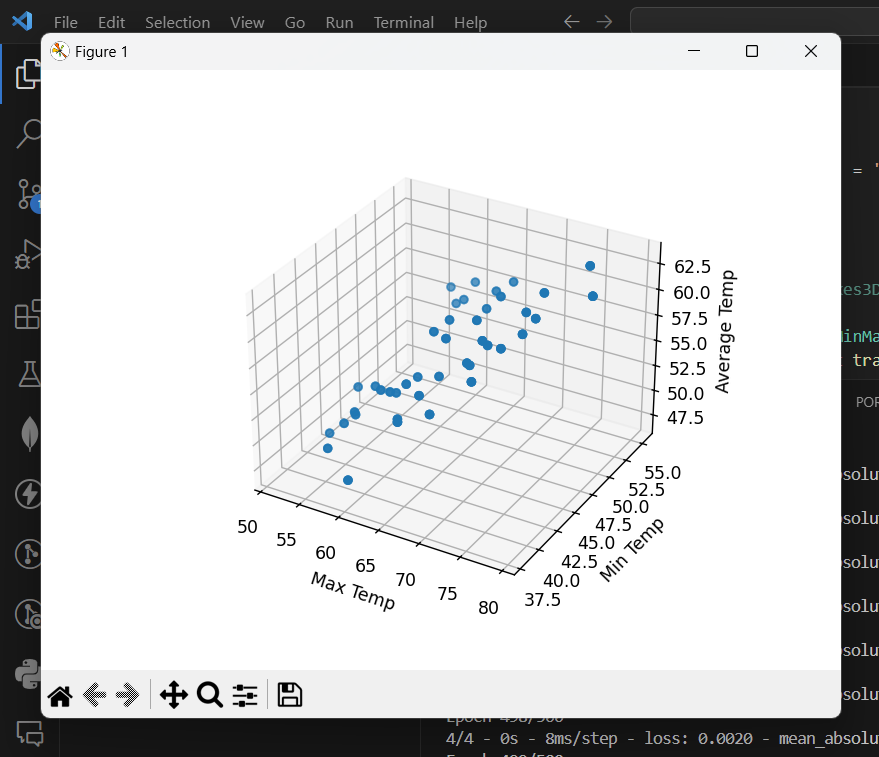
## CHAPTER 4

### Effective Preprocessing Enhances Model Performance (Learning Outcome)

**Outcome**: Proper data preprocessing and normalization significantly improve the performance of machine learning models.

**Results and Discussion**:

* **Preprocessing Steps**: In this project, we used MinMaxScaler to normalize both the input features (Tmax and Tmin) and the target variable (Tavg). This step ensured that all features were on a similar scale, which is crucial for the convergence of neural networks.
* **Impact on Model**: The normalization helped the neural network learn effectively, as indicated by the mean absolute error metric used during the training process. Without normalization, the model might have struggled with convergence and produced less accurate predictions.
* **Visualization**: The scatter plot of real vs. predicted temperatures shows that the predictions are reasonably close to the actual values, demonstrating the effectiveness of the preprocessing steps.



**Fig 4.1: MinMaxScaler Output**

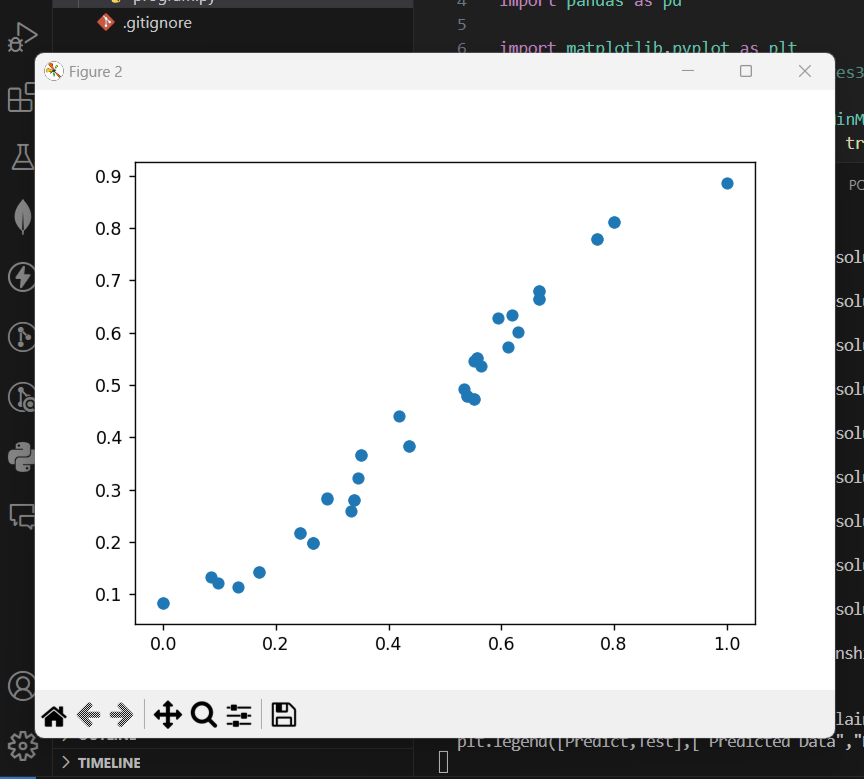
## CHAPTER 5

### Neural Network Architecture and Training Efficacy (Learning Outcome)

**Outcome**: A well-structured neural network with appropriate layers and activation functions can accurately predict weather parameters.

**Results and Discussion**:

* **Model Architecture**: The neural network model consisted of two hidden layers with 32 neurons each and ReLU activation functions, followed by an output layer with a sigmoid activation function. This architecture was chosen to capture the non-linear relationships in the weather data.
* **Training Process**: The model was trained for 500 epochs with a batch size of 32 using the RMSprop optimizer and mean squared error loss function. The training process showed a consistent decrease in the loss function, indicating that the model was learning effectively.
* **Performance**: The scatter plot of predicted vs. actual temperatures and the plot comparing real and predicted values over the test set both demonstrate that the model can predict temperatures with reasonable accuracy.



**Fig 5.1: Neural Network Output**

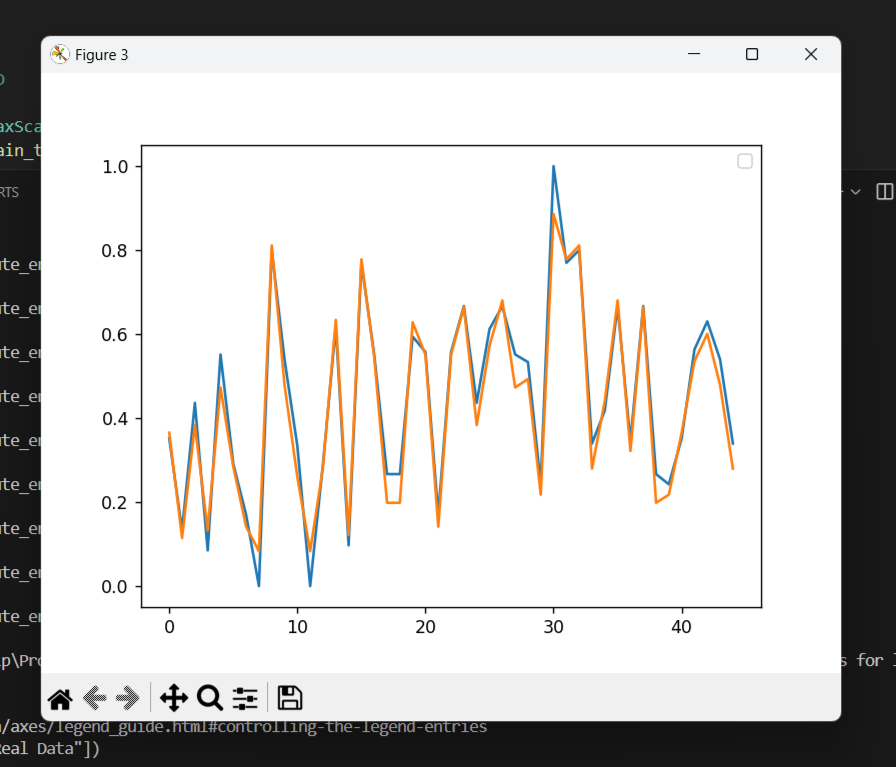
## CHAPTER 6

### Visualization as a Diagnostic Tool (Learning Outcome)

**Outcome**: Visualization techniques are crucial for diagnosing model performance and understanding the data.

**Results and Discussion**:

* **3D Scatter Plot**: The initial 3D scatter plot of Tmax, Tmin, and Tavg provided a visual understanding of the relationships between these variables. It helped confirm the appropriateness of using Tmax and Tmin as features to predict Tavg.
* **Scatter Plot of Predictions**: The scatter plot of predicted vs. actual values allowed for quick identification of the model's prediction accuracy. Points close to the diagonal line indicate accurate predictions, while deviations highlight areas where the model could be improved.
* **Comparison Plot**: The comparison plot of real vs. predicted values over the test set gave a temporal perspective of the model's performance, showcasing how well the model tracks actual temperature trends.



**Fig 6.1: Visualization Output**

## CHAPTER 7

### Summary

This weather prediction model leverages Python and several prominent libraries—Keras, Scikit-learn, NumPy, and Pandas—to forecast average temperatures using historical weather data from Los Angeles. The dataset includes daily maximum and minimum temperatures, which serve as input features to predict the average temperature.

Data preprocessing is conducted using Pandas for data manipulation and NumPy for array operations. Visualization of the data is achieved through Matplotlib, providing a 3D scatter plot to illustrate the relationship between maximum, minimum, and average temperatures. The dataset is normalized using Scikit-learn's MinMaxScaler to improve the model's training efficiency and performance.

The neural network model is constructed using Keras with a Sequential architecture. It consists of two hidden layers, each with 32 neurons and ReLU activation functions, and an output layer with a sigmoid activation function. The model is compiled with the mean squared error loss function and the RMSprop optimizer, tracking mean absolute error as a performance metric.

The model is trained on 70% of the data, with 30% reserved for testing, over 500 epochs with a batch size of 32. Post-training, the model's predictions are evaluated and visualized against the actual test data, demonstrating the model's ability to predict average temperatures based on the provided inputs. This approach highlights the efficacy of using deep learning techniques for weather prediction tasks.