**About IBM?**

IBM (International Business Machines) is a global technology company specializing in cloud computing, artificial intelligence, data analytics, and enterprise solutions, known for its innovation and leadership in IT services.

IBM was founded by Charles Ranlett Flint on June 16, 1911, as the Computing-Tabulating-Recording Company (CTR). It was later renamed International Business Machines (IBM) in 1924.

Experience of internship

**Before the Internship**

* **Excitement and Curiosity:** I was excited about the opportunity to apply my Python programming skills to a real-world project. I was curious about the challenges I would face and eager to learn new technologies and methodologies.
* **Preparation:** I spent time reviewing Python concepts, familiarizing myself with libraries related to data analysis and weather prediction, and reading about the company’s background and the projects they were working on.

**During the Internship**

* **Initial Learning Curve:** The beginning was a bit overwhelming as I had to quickly adapt to the company’s workflow and tools. However, the structured onboarding process and supportive team made the transition smoother.
* **Active Collaboration:** I actively collaborated with my mentor and team members, participating in meetings, code reviews, and discussions. This helped me understand the project requirements and align my work with the team’s objectives.
* **Excitement and Curiosity:** I was excited about the opportunity to apply my Python programming skills to a real-world project. I was curious about the challenges I would face and eager to learn new technologies and methodologies.
* **Preparation:** I spent time reviewing Python concepts, familiarizing myself with libraries related to data analysis and weather prediction, and reading about the company’s background and the projects they were working on

**After the Internship**

* **Reflection and Growth:** Reflecting on the internship, I realized how much I had grown both technically and professionally. I gained a deeper understanding of machine learning, data processing, and model evaluation.
* **Confidence:** Successfully working on a real-world project boosted my confidence in my abilities to tackle complex problems and contribute to a team.
* **Career Aspirations:** The experience solidified my interest in data science and machine learning, and I became more enthusiastic about pursuing a career in this field.

**Technologies and Tools Used**

**NumPy** is a general-purpose array-processing package.

**Pandas** has functions for analyzing, cleaning, exploring, and manipulating data.

**Matplotlib** is an amazing visualization library in **Python**for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays

**Keras** is a high-level, user-friendly API used for building and training neural networks.

**Sklearn**

**preprocessing**: Contains tools for preprocessing data, such as scaling.

**model\_selection**: Used for splitting data into training and test sets.

**L A\_Weather.csv**

The CSV file titled "L A\_Weather.csv" contains 149 entries with six columns:

'STATION', 'NAME', 'DATE', 'TAVG' (Average Temperature), 'TMAX' (Maximum Temperature), and 'TMIN' (Minimum Temperature).

The dataset records weather data for Los Angeles over different dates, with the temperatures provided in degrees Fahrenheit.

Program Explanation

Figure 1

Code

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)

* **Purpose**: This figure is a 3D scatter plot showing the relationship between the maximum temperature (Tmax), minimum temperature (Tmin), and average temperature (Tavg).
* **Axes**:
  + X-axis: Maximum Temperature
  + Y-axis: Minimum Temperature
  + Z-axis: Average Temperature
* **Insight**: Visualizes how these three-temperature metrics are related in 3D space.

Explanation of the Neural Network Model

Code

Temp = np.concatenate([Tmax,Tmin],axis=0)

Temp = np.transpose(Temp)

Tavg = np.transpose(Tavg)

scaler = MinMaxScaler()

scaler.fit(Temp)

Temp = scaler.transform(Temp)

scaler1 = MinMaxScaler()

scaler1.fit(Tavg)

Tavg = scaler1.transform(Tavg)

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

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)

predict = model.predict(X\_test,verbose=1)

**Data Preparation:**

* **Concatenate and Transpose:** Combines maximum and minimum temperatures into a single array Temp, and transposes both Temp and average temperatures (Tavg).
* **Normalization:** Uses MinMaxScaler to normalize Temp and Tavg values between 0 and 1.
* **Train-Test Split:** Splits the normalized data into training and testing sets, with 70% for training and 30% for testing.

**Model Building:**

* **Sequential Model:** Constructs a Sequential (appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor) neural network model.
* **Layers:** Adds two dense (fully connected) layers with 32 neurons each and ReLU (rectified linear unit, is a non-linear activation function used for deep neural networks in machine learning) activation, followed by an output layer with 1 neuron and sigmoid activation.

**Model Compilation:**

* **Loss Function:** Uses mean squared error to measure the model's performance.
* **Optimizer:** Utilizes the RMSprop optimizer for training.
* **Metrics:** Tracks mean absolute error during training.

**Model Training:**

* **Fit:** Trains the model on the training data for 500 epochs (**feeding the entire training dataset through the network once. During this process, the network's weights and biases are adjusted to minimize the error between the predicted and actual outputs.)** with a batch size (**the number of samples that you feed into your model at each iteration of the training process**) of 32.

**Prediction:**

* **Predict:** Generates predictions on the test set and stores them in predict.

**Figure 2: Predicted vs Actual Scatter Plot**

**Code**

plt.figure(2)

plt.scatter(Y\_test, predict)

plt.show(block=False)

* **Purpose**: This figure is a scatter plot comparing the predicted average temperatures (predict) to the actual average temperatures (Y\_test).
* **Axes**:
  + X-axis: Actual Average Temperature
  + Y-axis: Predicted Average Temperature
* **Insight**: This plot helps visualize the performance of the neural network model. Ideally, points should lie along the line y=x, indicating perfect predictions.

**Figure 3: Predicted vs Actual Line Plot**

**Code**

plt.figure(3)

Test = plt.plot(Y\_test)

Predict = plt.plot(predict)

plt.legend([Predict, Test], ["Predicted Data", "Real Data"])

plt.show()

* **Purpose**: This figure plots the predicted temperatures and the actual temperatures over the test dataset.
* **Insight**: By comparing the two plots, you can see how closely the predicted values match the actual values over the range of the dataset.