

**Final Project: Recommendation System**

Vivek Kandel

Chandana Kandari



February 17, 2021

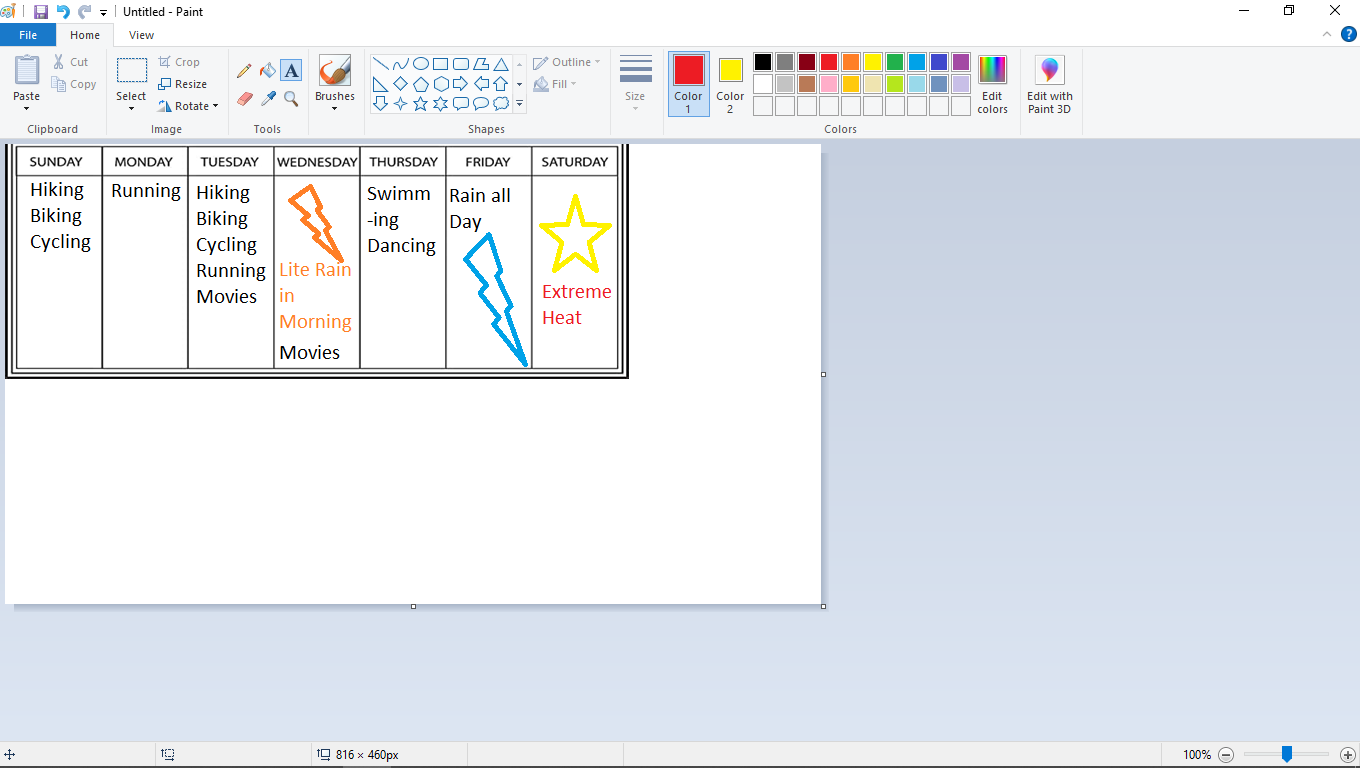
ASU: IFT-598: Advance Analytics for Big data / AI

Prof. Rucker

**Recommendation System: Best Activity for the Week**

**Introduction:**

Human lives revolve around various environmental factors, so planning to do something like hiking, biking, running, or going outside the house depends on weather. Throughout the history, we have learned to improve the weather forecast, however, we have not figured out the best way to utilize it properly. For the project, we are planning to create a recommendation system for users, so they can decide what would be the best day to go outside the house for specific activities. Overall, we want to utilize the weather features to decide if certain activity is suitable for that day/week or not.



***Figure 1: Example Output GUI for users***

Users would be able to get recommendation for the whole week as shown in figure1 and it would be based on the forecasted weather from API as inputs for the prediction model. The specific activities will be recommended based on the multiple-class classification algorithm. And the extreme warning like rain or extreme heat will be explicitly coded (using if else based on API’s minimum/max temperature, amount of rain and wind) instead of using the machine learning algorithm. The recommendation is based on the training data, so we will try to include as many activities as possible in the training dataset. Furthermore, we would also explore to see if we can make this self-learning based on user feedback or generated new testing data based on user feedback.

**The questions my work will answer includes:**

1. How does the probability of getting certain activity (near 1) vs. (near 0) change for every weather feature like temperature, rain, and wind?

We would investigate the Neuron’s weights and bias that results in specific activity to see which Neuron has the largest impact on the prediction. This would show me how each weather features impact probability of getting certain activity.

1. How much computation power would it take to make the prediction?

It depends on the size of training data, however, after the prediction model is created then using the model is quick. Furthermore, the response time of the weather API from AccuWeather also matters because it is used as the input for the prediction model. We have decided not to use prediction model for extreme weather cases like for rain, or extreme heat by using conditional statement where it warns the user not to plan for outdoor activity.

1. How is the recommendation model evaluated and validated?

The data is going to be split into training and testing. Testing data is going to be used to evaluate the accuracy of the prediction model. We would specifically use confusion matrix to evaluate my testing prediction. The negative consequence of this prediction model that recommends specific activity is low and it can be used as guidance.

1. What is the benefit for the user who utilized this prediction model?

Conventionally, a typical user checks the weather forecast manually before making plans to do certain activity, so this will save them time and help them make proper decision. And it can be also useful for users who do not consider or forget about weather factors that impact their activity.

1. What other factors impact the recommended activity or what are the limitation?

List of activities for the day will be recommended, but it is up to the user to decide which activity to choose. We could sort the recommended activity based on the higher threshold/probability. However, the predictive models would be limited to certain activities and if the user wants the recommendation for more activities, they will not be able to get it because the activities are limited to activities in training dataset.

**Workflow:**

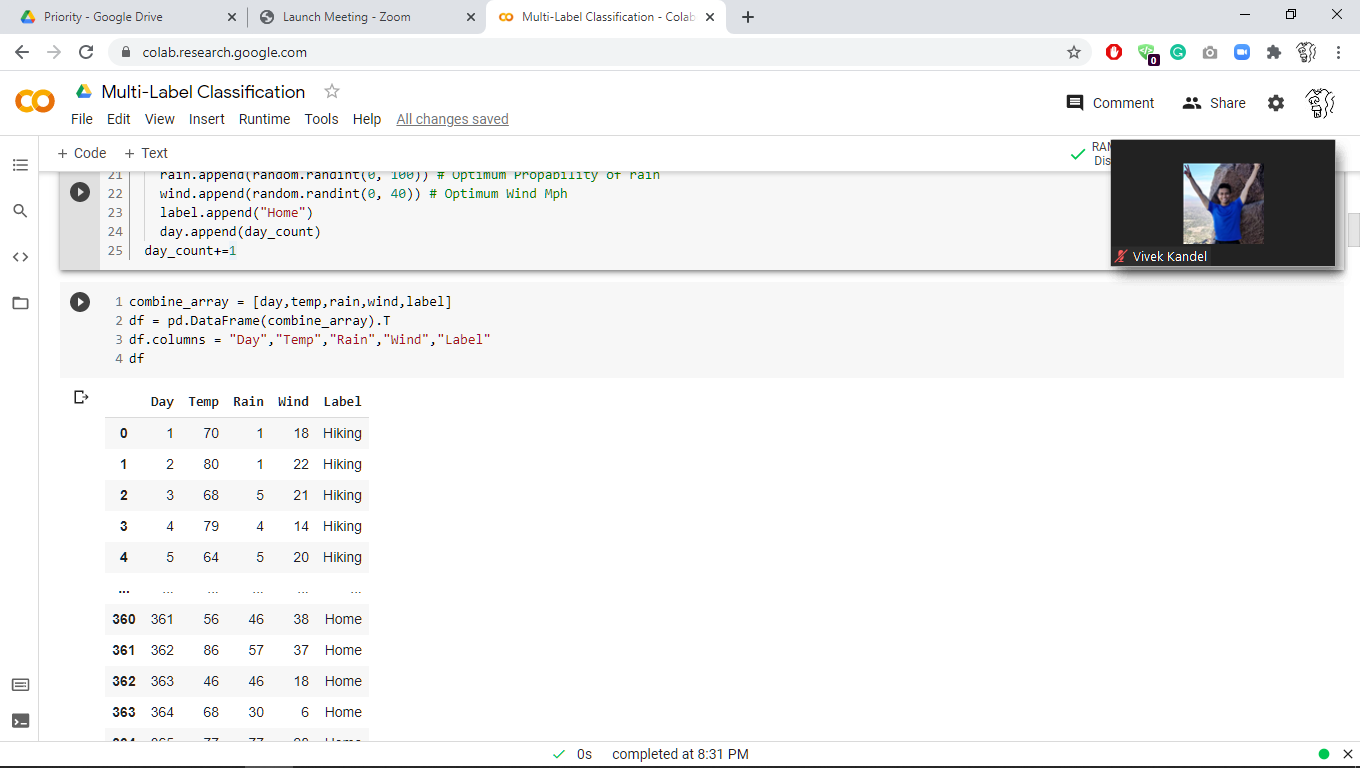
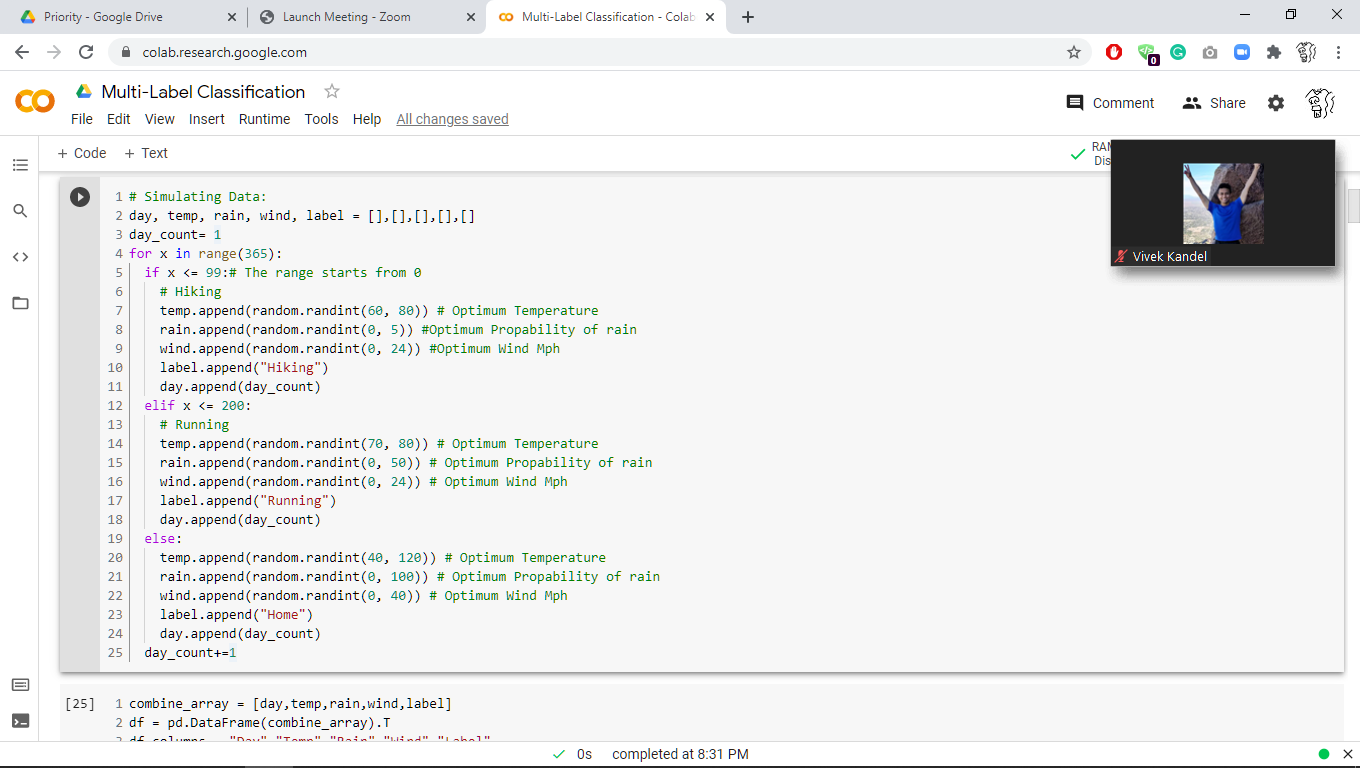
Simulate the data with features (temperature, wind, rain) and label (activity). Analyze the data and check for abnormality and missing data.

Define the data Schema and prepare the data structure (data type and split to training and testing) to feed into Neural Network Model for training and testing.

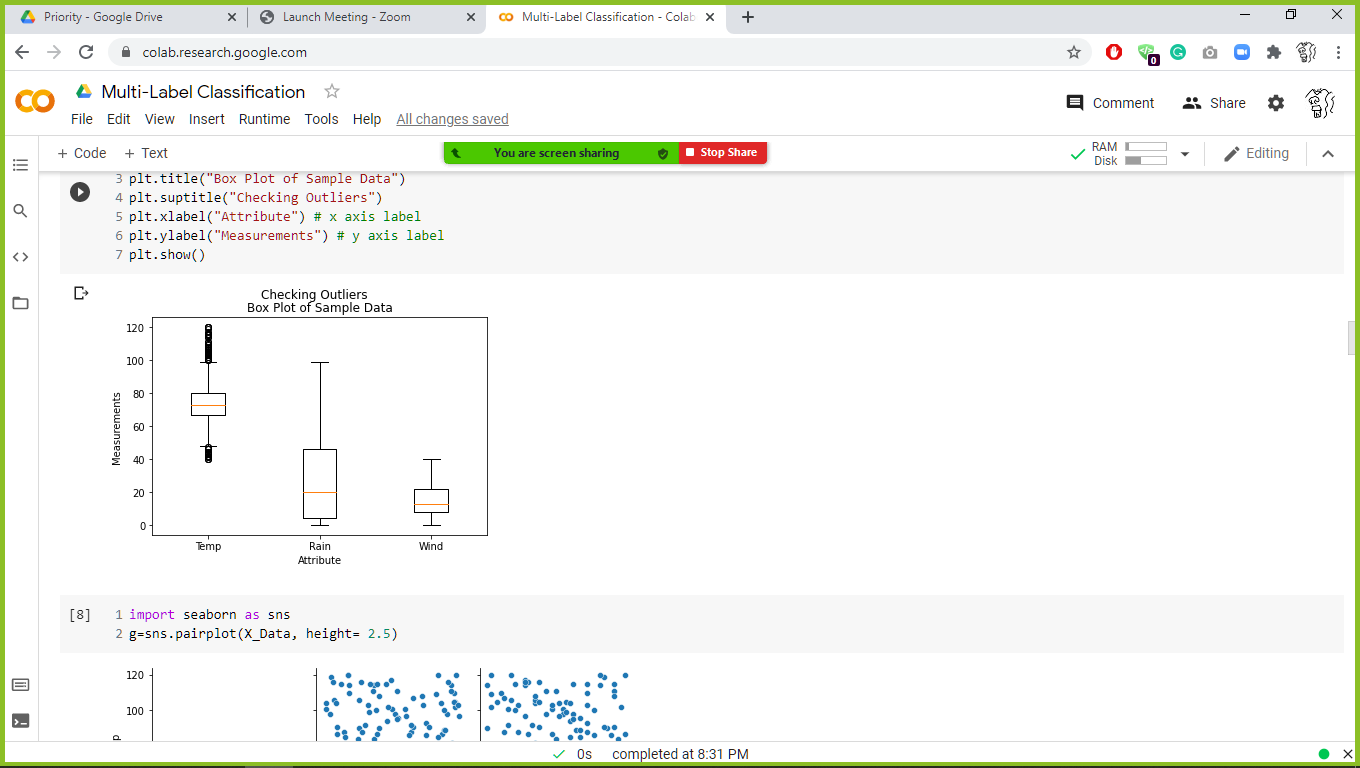
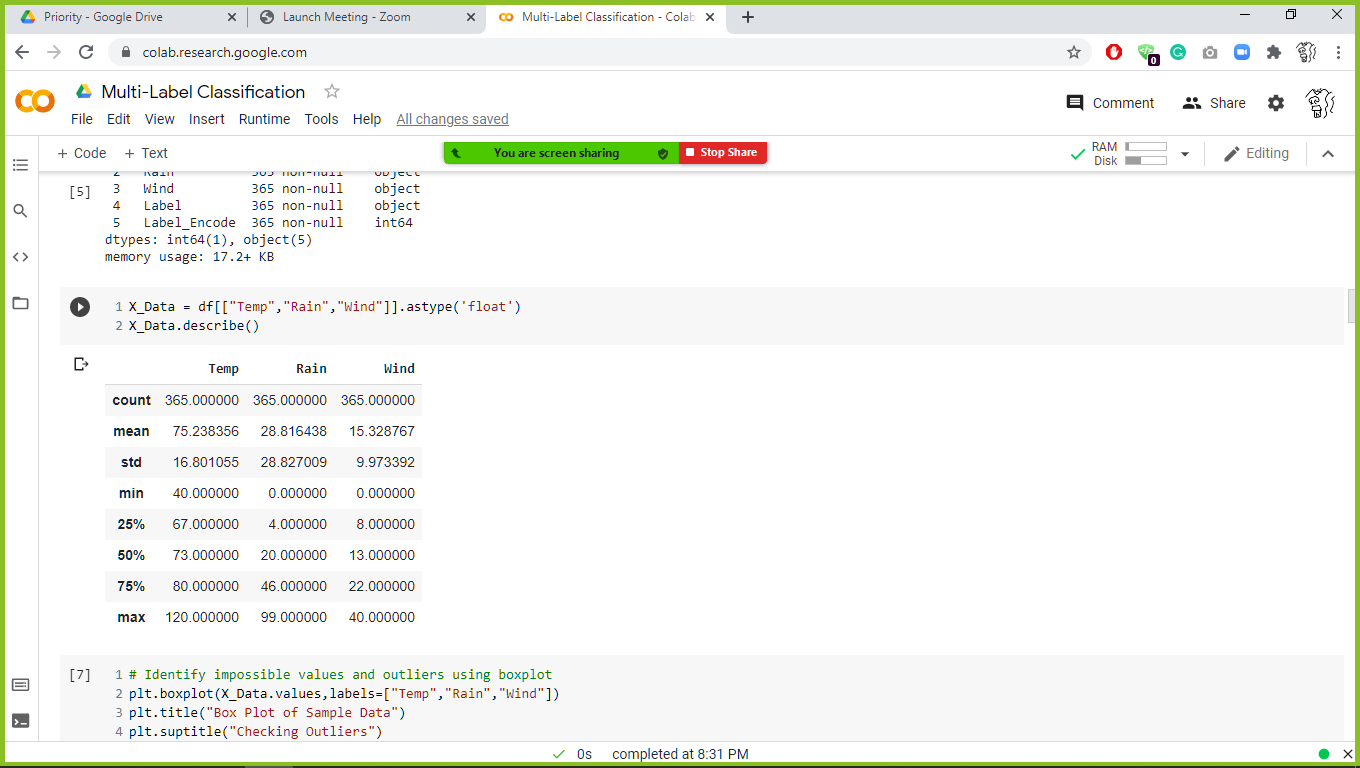
Interpret/Analyze the results (Predictions for testing data). Find the ways to optimize the error using confusion matrix.

Deploy the predictive model to recommend activity using the API from AccuWeather as input for the predictive model.

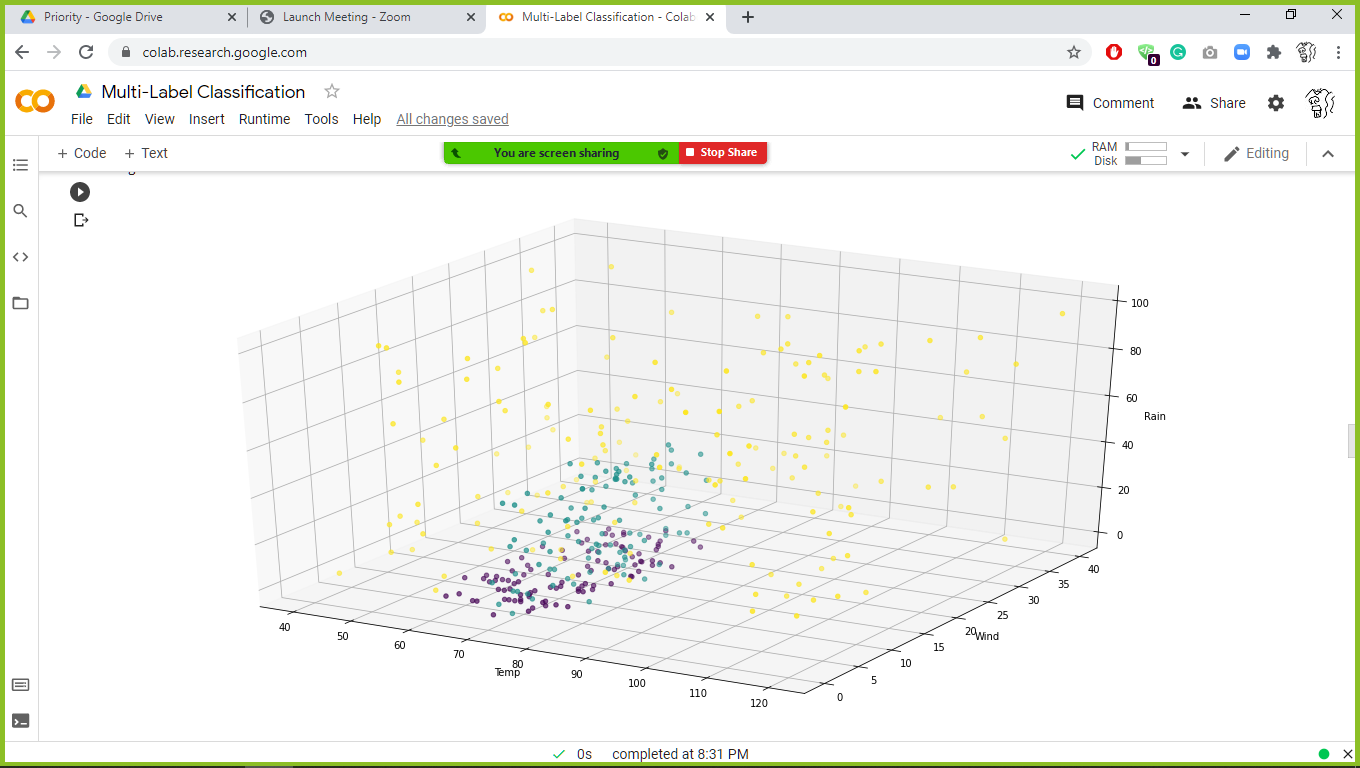
**Sample Data:** We will be simulating the data as our sample data using Python and we specifically choose three features because it is easy to visualize them.



**Data Analysis and Visualization:**



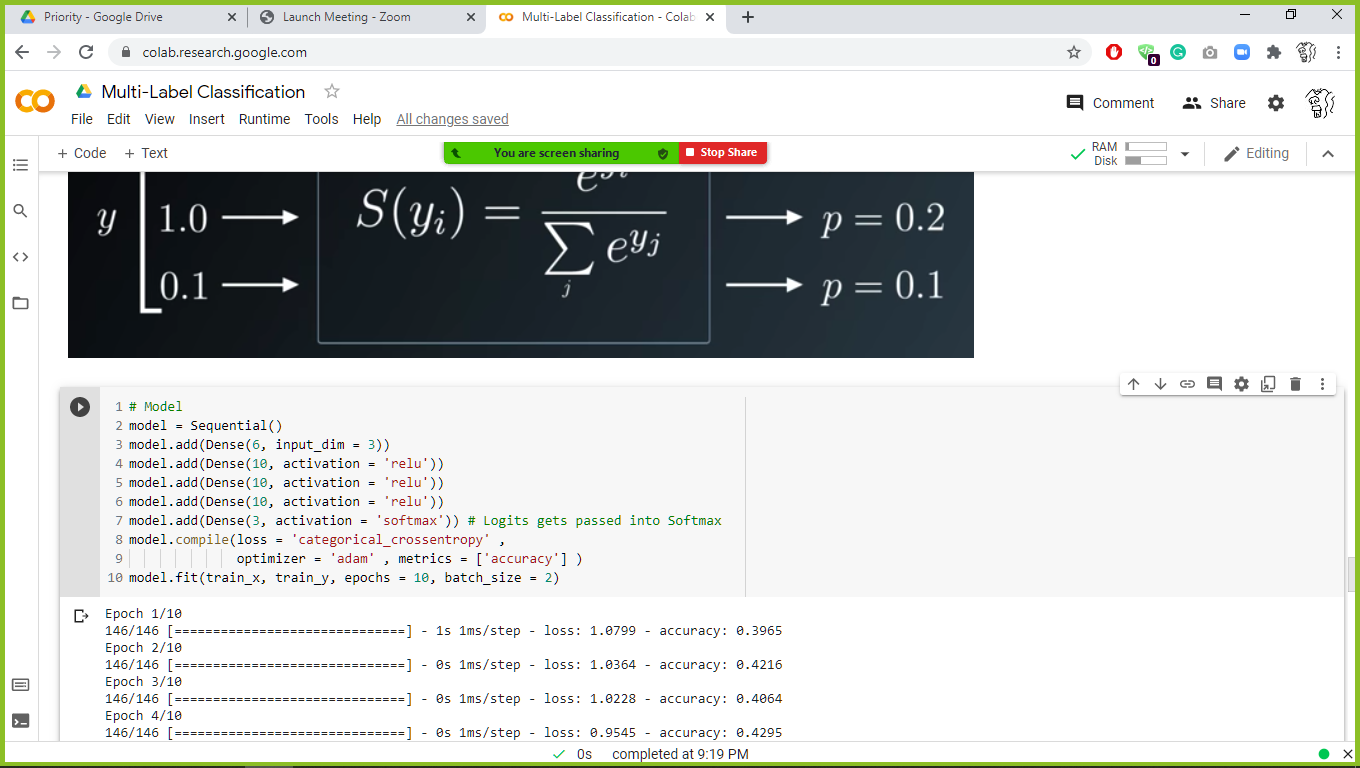
We were able to get basic statistics using Pandas describe function (syntax: df.describe()) and then visualized the weather features using various visualization libraries.



Three-dimension graph gave a dynamic approach and made the data more interactive. As we can see in the diagram above, the labels are clearly impacted by the weather features, however, some labels overlap.

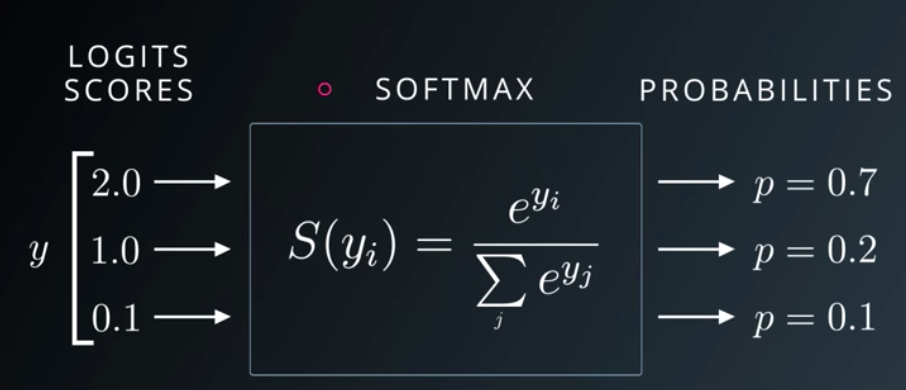
**Model:**

We will be using Multi-Class Classification using Artificial Neural Network (Multi-Layered Perceptron). Each label (activity) is going to have its own probability that results between 0 to 1. And finally for each label, if the final value for prediction is greater or equal to 0.5, then that label (activity) will be recommended.

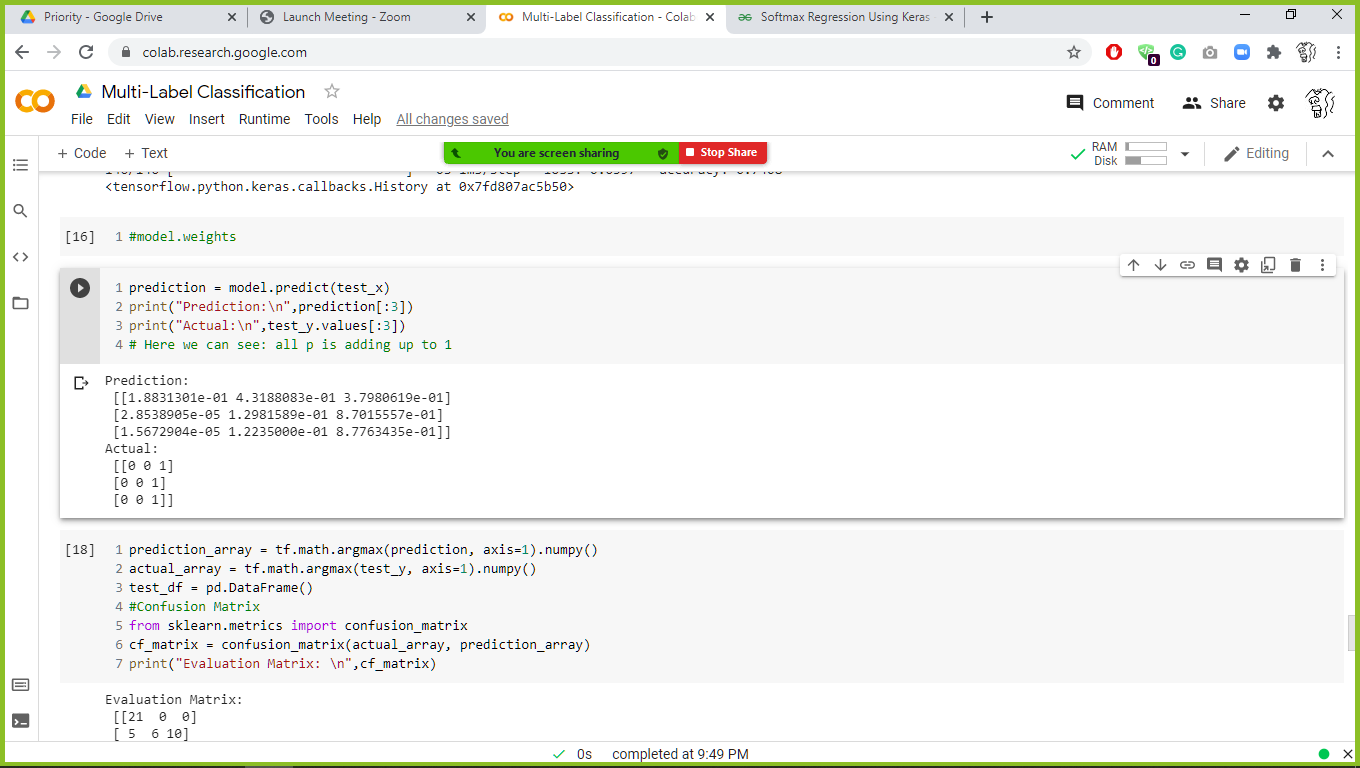


***• Sequential:*** That defines a SEQUENCE of layers in the neural network.• ***Dense:*** Adds a layer of neurons with specific ***neurons***.• Each layer of neurons needs an ***activation function*** to account for ***non-linearity***.

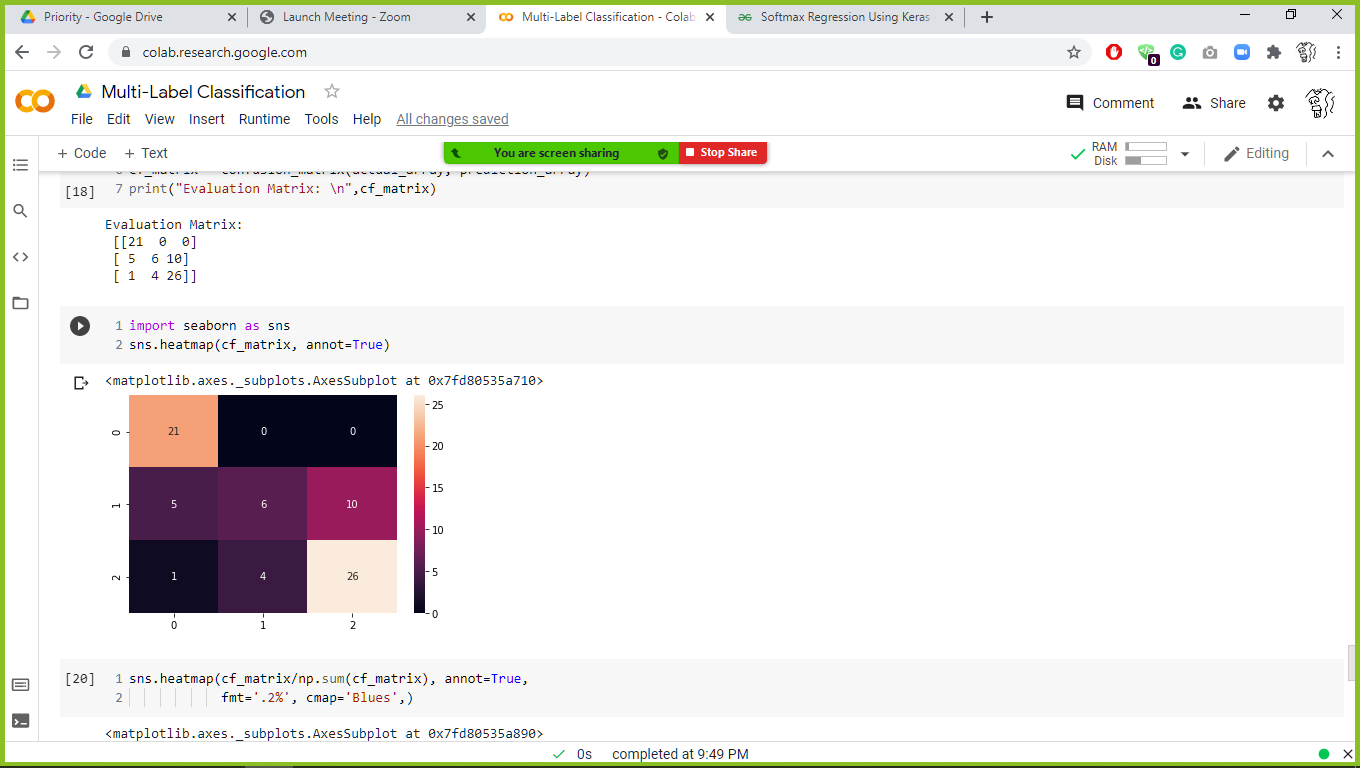
• ***Relu (activation function)***: Effectively means “If X > 0 return X, else return 0″

• ***SoftMax:*** It is used at the output layer as an activation function because there are three labels, and it normalizes the logits in the output layers into the range (Probabilities) 0 to 1.

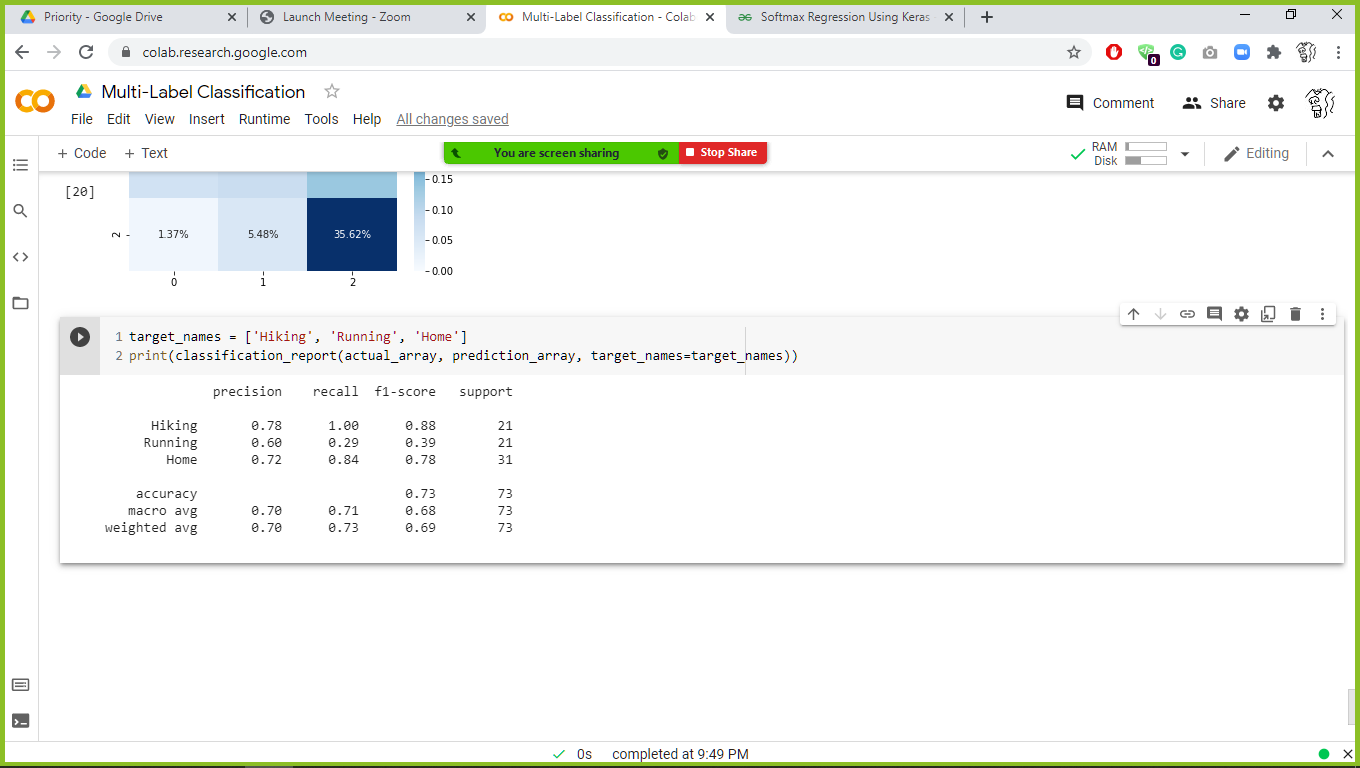
**Output and Evaluation of the model:**



After the training was done, we were able to predict and evaluate the model using testing data (syntax: model.predict(test\_df)).



**Confusion Matrix and Evaluation:**



**CRISP Comment:**

We have referred CRISP DM Framework for Data Science and Machine Learning from LinkedIn and found that we have followed almost every step that is stated in the framework.

The first phase is **Business Understanding**, this step demands the developer to identify the goal and frame the business problem (figuring out the best way to utilize weather features as well as weather forecast for deciding on a Physical activity that is suitable based on it), gather information (get sample data from NOAA for training and use API to predict the activity), prepare analytical goal (improve the activity recommendation), and lastly, a pictorial representation (Figure 1: Example Output GUI for users).

**Data Understanding** phase of CRISP DM Framework focus on collecting the data (Training and Test dataset), describing (features like Average wind, Evaporation, Precipitation, Maximum Temperature and Minimum Temperature and label describing Physical activities) and exploring the data involves analyzing the data in hand for Dependent and Independent Variable Identification (Weather features are independent variables and label describing Physical activities is dependent variable).

**Data preparation** phase prepares and cleans the provided data making it ready for modeling. Our data is mostly cleaned that is it is free from missing values, null values, NA’s. But we are performing **Feature Engineering** to create a label attribute (Physical activities) for the training dataset from the available weather feature attributes (features like Average wind, Evaporation, Precipitation, Maximum Temperature and Minimum Temperature).

Once the above steps are done, we are ready for **Modeling** phase by providing this Training and Test dataset to the prediction model for recommendations.

For **Evaluation of the Model,** we will be using the **Confusion Matrix** to calculate the accuracy and precision.

In **Deployment** phase**,** once the model is created and tested and evaluated on the Test and Validation data, this is presented to the IFT 598 via Zoom session recording.

**Infrastructure components:**

* **Python** comes with many libraries. Many of these inbuilt libraries are for Machine Learning and Artificial Intelligence and can easily be applied out of the box.For example, scikit-learn is usedfor data mining, analysis, and Machine Learning.
* **Scikit-learn** provides simple and efficient tools for predictive data analysis and it is Accessible to everybody, and reusable in various contexts. It is Built on NumPy, SciPy, and matplotlib.
* **Pandas** library provides high-performance, easy-to-use data structures and data analysis tools for the python programming language.
* The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It supports data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
* **Keras** is an open-source, ML library that wraps around the functionalities of other ML and DL libraries, including TensorFlow. Keras works with Pandas datasets, creating Tensors. Many users and data scientists like using Keras because it makes TensorFlow far less prone to make models that offer the wrong conclusions.

**Conclusion:**

We wanted to utilize the weather data to make better decisions for our activity in the future. Depending on the accuracy of the model, we can decide to deploy the model and serve the users. We think this project will save people time by helping them make better decisions. The best place to implement this predictive model will in smartphone because it is easily accessible, and notifications can easily reach to the user compare to web application or software in PC. We also think that weather has various other impact in our lives and finding the relationship between them can help us live in harmony.

**Source Code:**

# -\*- coding: utf-8 -\*-  
*"""Multi-Label Classification  
"""*import os  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import confusion\_matrix,classification\_report  
from matplotlib import pyplot as plt  
import random  
import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
print("TensorFlow version: {}".format(tf.\_\_version\_\_))  
  
# Simulating Data:  
day, temp, rain, wind, label = [],[],[],[],[]  
day\_count= 1  
for x in range(365):  
 if x <= 99:# The range starts from 0   
 # Hiking  
 temp.append(random.randint(60, 80)) # Optimum Temperature   
 rain.append(random.randint(0, 5)) #Optimum Propability of rain  
 wind.append(random.randint(0, 24)) #Optimum Wind Mph  
 label.append("Hiking")  
 day.append(day\_count)  
 elif x <= 200:  
 # Running  
 temp.append(random.randint(70, 80)) # Optimum Temperature   
 rain.append(random.randint(0, 50)) # Optimum Propability of rain  
 wind.append(random.randint(0, 24)) # Optimum Wind Mph  
 label.append("Running")  
 day.append(day\_count)   
 else:  
 temp.append(random.randint(40, 120)) # Optimum Temperature   
 rain.append(random.randint(0, 100)) # Optimum Propability of rain  
 wind.append(random.randint(0, 40)) # Optimum Wind Mph  
 label.append("Home")  
 day.append(day\_count)  
 day\_count+=1  
  
combine\_array = [day,temp,rain,wind,label]  
df = pd.DataFrame(combine\_array).T  
df.columns = "Day","Temp","Rain","Wind","Label"  
df["Label\_Encode"] = df["Label"].map({"Hiking":0,"Running":1,"Home":2})  
  
df.info()  
  
X\_Data = df[["Temp","Rain","Wind"]].astype('float')  
X\_Data.describe()  
  
# Identify impossible values and outliers using boxplot  
plt.boxplot(X\_Data.values,labels=["Temp","Rain","Wind"])  
plt.title("Box Plot of Sample Data")  
plt.suptitle("Checking Outliers")  
plt.xlabel("Attribute") # x axis label  
plt.ylabel("Measurements") # y axis label  
plt.show()  
  
import seaborn as sns  
g=sns.pairplot(X\_Data, height= 2.5)  
  
plt.scatter(df['Temp'],  
 df['Wind'],  
 c=df["Label\_Encode"].values)  
plt.xlabel("Temp")  
plt.ylabel("Wind")  
plt.show()  
  
from mpl\_toolkits.mplot3d import Axes3D  
import matplotlib.pyplot as plt  
fig = plt.figure()  
fig = plt.figure(figsize=(20,10))  
ax = fig.add\_subplot(111, projection='3d')  
x =temp  
y =wind  
z =rain  
ax.scatter(x, y, z, c=df["Label\_Encode"], marker='o')  
ax.set\_xlabel('Temp')  
ax.set\_ylabel('Wind')  
ax.set\_zlabel('Rain')  
plt.show()  
  
# Normalizing data = Helps in Conversion in SGD  
X\_Data = X\_Data.apply(lambda x:( (x - x.min()) / (x.max()-x.min())))  
  
Y\_Data = df["Label\_Encode"]  
Y\_Data = pd.get\_dummies(Y\_Data)  
  
train\_x, test\_x, train\_y, test\_y = train\_test\_split(X\_Data, Y\_Data, test\_size=0.20, random\_state=40)  
print(train\_x.shape); print(train\_y.shape)  
  
pd.concat([train\_x,train\_y],axis=1)  
  
"""\*\*# SoftMax a.k.a Normalization over different classes\*\*  
<img src="https://miro.medium.com/max/906/1\*\_IDMoFnoJT916hhUREiFAQ.png" alt="Girl in a jacket" style="vertical-align:top">  
"""  
  
# Model  
model = Sequential()  
model.add(Dense(6, input\_dim = 3))  
model.add(Dense(10, activation = 'relu'))  
model.add(Dense(10, activation = 'relu'))  
model.add(Dense(10, activation = 'relu'))   
model.add(Dense(3, activation = 'softmax')) # Logits gets passed into Softmax  
model.compile(loss = 'categorical\_crossentropy' ,  
 optimizer = 'adam' , metrics = ['accuracy'] )  
model.fit(train\_x, train\_y, epochs = 10, batch\_size = 2)  
  
#model.weights  
  
prediction = model.predict(test\_x)  
print("Prediction:\n",prediction[:3])  
print("Actual:\n",test\_y.values[:3])  
# Here we can see: all p is adding up to 1  
  
prediction\_array = tf.math.argmax(prediction, axis=1).numpy()  
actual\_array = tf.math.argmax(test\_y, axis=1).numpy()  
test\_df = pd.DataFrame()  
#Confusion Matrix  
from sklearn.metrics import confusion\_matrix  
cf\_matrix = confusion\_matrix(actual\_array, prediction\_array)  
print("Evaluation Matrix: \n",cf\_matrix)  
  
import seaborn as sns  
sns.heatmap(cf\_matrix, annot=True)  
  
sns.heatmap(cf\_matrix/np.sum(cf\_matrix), annot=True,   
 fmt='.2%', cmap='Blues',)  
  
target\_names = ['Hiking', 'Running', 'Home']  
print(classification\_report(actual\_array, prediction\_array, target\_names=target\_names))

**References:**

Figure1- Days in a Week:

<https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.amazon.com%2FWall-Pops-WPE98895-Whiteboard-Calendar%2Fdp%2FB005S0GKX8&psig=AOvVaw2kABpJ2eR9FirzjeUT93Yt&ust=1614810754856000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCJD25f3Uku8CFQAAAAAdAAAAABAH>

CRISP – Framework:

[(8) Chapter 1 - Introduction to CRISP DM Framework for Data Science and Machine Learning | LinkedIn](https://www.linkedin.com/pulse/chapter-1-introduction-crisp-dm-framework-data-science-anshul-roy/)

Getting Started with Keras

<https://www.geeksforgeeks.org/softmax-regression-using-keras/>