Applied Tech. Project 81 - Multi-Class Classification II

March 26, 2021

0.1 Instructions

Goal of the Project This project is designed for you to practice and solve the activities that are based on the concepts covered in the following lessons:

1. Logistic Regression - Univariate Classification I
2. Logistic Regression - Decision Boundary
3. Logistic Regression - Multiclass Classification I
4. Logistic Regression - Multiclass Classification II

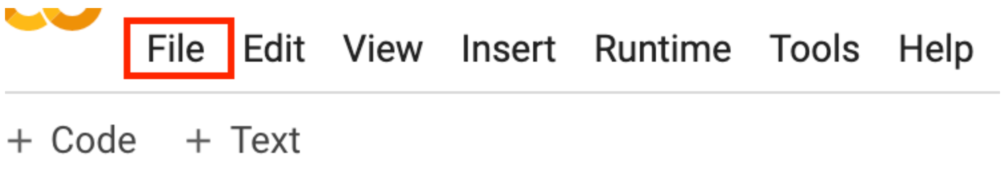
Getting Started:

1. Click on this link to open the Colab file for this project.

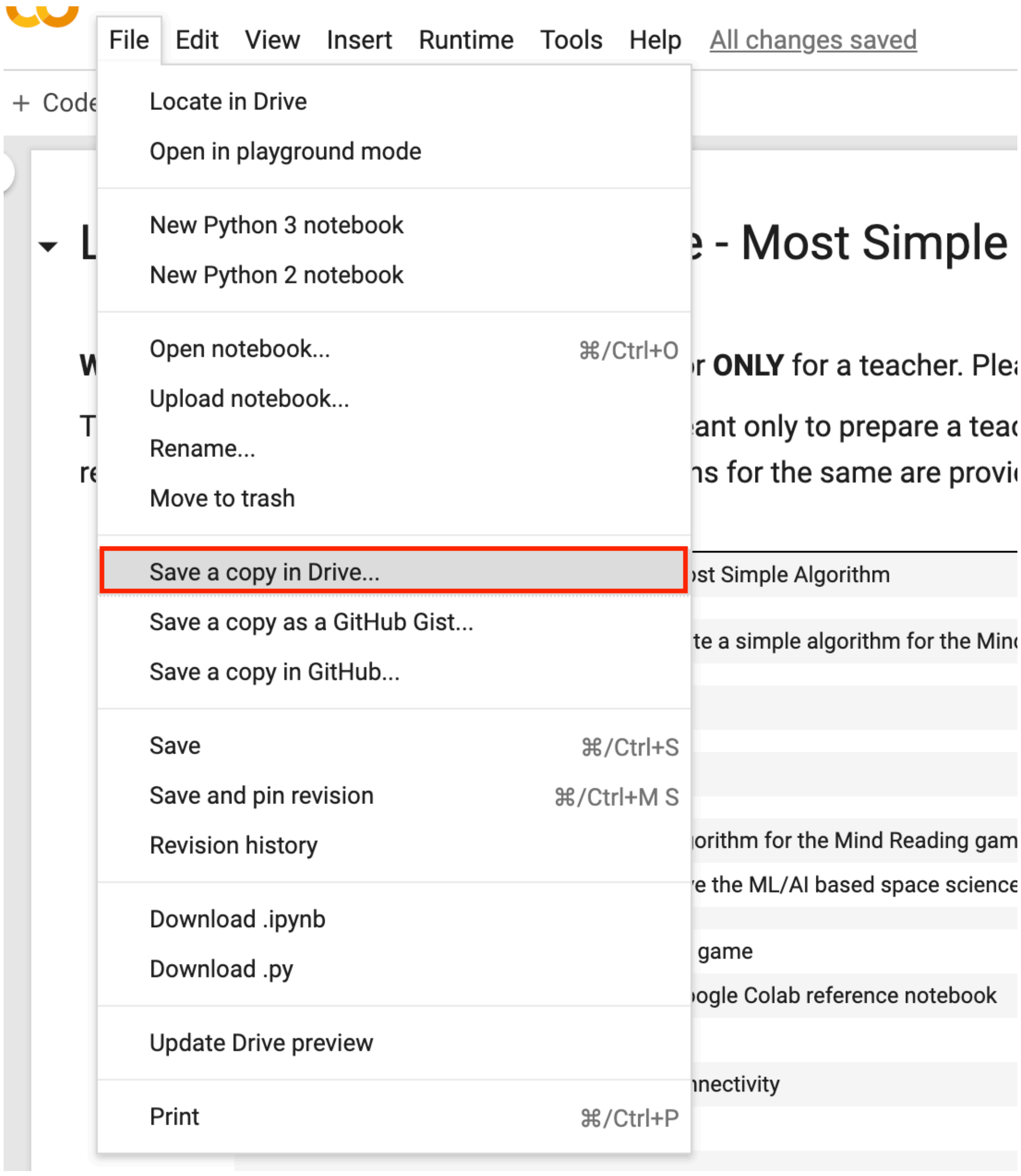
   https://colab.research.google.com/drive/1H4TAhvov035k7DNHBBRiUwd2VHiT0QaL

2. Create a duplicate copy of the Colab file as described below.

- Click on the **File menu**. A new drop-down list will appear.



- Click on the **Save a copy in Drive** option. A duplicate copy will get created. It will open up in the new tab on your web browser.



3. After creating the duplicate copy of the notebook, please rename it in the **YYYY-MM-DD_StudentName_Project80** format.
4. Now, write your code in the prescribed code cells.

0.1.1 Problem Statement

In this project, you are going to create your synthetic data for multiclass classification, synthesize the data for training, and create a Random Forest Classification model and a Logistic Regression model to classify the data using Machine Learning. Conclude the project by comparing the models and their results.

0.1.2 List of Activities

Activity 1: Create the Dummy Dataset

Activity 2: Dataset Inspection

Activity 3: Train-Test Split

Activity 4: Apply SMOTE

Activity 5: Random Forest Classification - Model Training

Activity 6: Random Forest Classification - Model Prediction and Evaluation (Testing Set)

Activity 7: Logistic Regression - Model Training

Activity 8: Logistic Regression - Model Prediction and Evaluation (Testing Set)

Activity 1: Create the Dummy Dataset In this activity, you have to create a dummy dataset for multiclass classification.

The steps to be followed are as follows:

1. Create a dummy dataset having two columns representing two independent variables and a third column representing the target.

The number of records should be divided into 6 random groups like `[500, 2270, 1900, 41, 2121, 272]` such that the target columns has 6 different labels `[0, 1, 2, 3, 4, 5]`.

Recall:

To create a dummy data-frame, use the `make_blob()` function of the `sklearn.datasets` module which will return two arrays `feature_array` and the `target_array`. The syntax for the `make_blob()` function is as follows:

Syntax: `make_blobs(n_samples, centers, n_features, random_state, cluster_std)`

```
[ ]: # Instruction to remove warning messages
```

```
[ ]: # Create two arrays using the 'make_blobs()' function and store them in the␣
     ↪'features_array' and 'target_array' variables.
```

Hint:

In the `make_blobs()` function use `n_samples=[500, 2270, 1900, 41, 2121, 272]` and `center=None` for the division of target label into seven groups.

---

2. Print the object-type of the arrays created by the `make_blob()` function and also print the number of rows and columns in them:

```
[ ]: # Find out the object-type of the arrays created by the 'make_blob()' function␣
     ↪and the number of rows and columns in them.

     # Print the type of 'features_array' and 'target_array'

     # Print the number of rows and column of 'features_array'

     # Print the number of rows and column of 'target_array'
```

Q: How many rows are created in the feature and target columns?

A:

---

3. Create a DataFrame from the two arrays using a Python dictionary.

Steps: (Learned in "Logistic Regression - Decision Boundary" lesson) - Create a dummy dictionary.

- Add the feature columns as keys `col 1`, `col 2`, and target column as the `target`.
- Add the values from the feature and target columns one by one respectively in the dictionary using List Comprehension.
- Convert the dictionary into a DataFrame
- Print the first five rows of the DataFrame.

```
[ ]: # Create a Pandas DataFrame containing the items from the 'features_array' and␣
     ↪'target_array' arrays.

     # Import the module

     # Create a dummy dictionary


     # Convert the dictionary into DataFrame

     # Print first five rows of the DataFrame
```

Hint:

Use function `from_dict()` to convert Python Dictionary to DataFrame.

Syntax: `pd.DataFrame.from_dict(some_dictionary)`

After this activity, the DataFrame should be created with two independent features columns and one dependent target column.

Activity 2: Dataset Inspection In this activity, you have to look into the distribution of the labels in the `target` column of the DataFrame.

1. Print the number of occurrences of each label in the `target` column:

```
[ ]: # Display the number of occurrences of each label in the 'target' column.
```

2. Print the percentage of the samples for each label in the `target` column:

```
[ ]: # Get the percentage of count of each label samples in the dataset.
```

Q: How many unique labels are present in the DataFrame? What are they?

A:

Q: Is the DataFrame balanced?

0.2 A:

3. Create a scatter plot between the columns `col 1` and `col 2` for all the labels to visualize the clusters of every label (or points):

```
[ ]: # Create a scatter plot between 'col 1' and 'col 2' columns separately for all␣
     ↪the labels in the same plot.

     # Import the module


     # Define the size of the graph

     # Create a for loop executing for every unique label in the `target` column.


       # Plot the scatter plot for 'col 1' and 'col 2' where 'target ==i"


     # Plot the x and y lables

     # Display the legends and the graph
```

Hint: Revise the lesson "Logistic Regression - Decision Boundary".

After this activity, the labels to be predicted that is the target variables and their distribution should be known.

Activity 3: Train-Test Split We need to predict the value of the `target` variable, using other variables. Thus, the `target` is the dependent variable and other columns are the independent variables.

1. Split the dataset into the training set and test set such that the training set contains 70% of the instances and the remaining instances will become the test set.

2. Set `random_state = 42`.

[ ]:
```
# Import 'train_test_split' module


# Create the features data frame holding all the columns except the last column
# and print first five rows of this dataframe

# Create the target series that holds last column 'target'
# and print first five rows of this series

# Split the train and test sets using the 'train_test_split()' function.
```

3. Print the number of rows and columns in the training and testing set:

[ ]:
```
# Print the shape of all the four variables i.e. 'X_train', 'X_test', 'y_train'␣
↪and 'y_test'
```

After this activity, the features and target data should be split into training and testing data.

Activity 4: Apply SMOTE In this activity, if the data is imbalanced, oversample the data for the minority classes in the following way:

1. Create an object for the SMOTE using `SMOTE()` function.
2. Synthesize the data for the minority class using `fit_sample()` function by passing the feature and target training variable.
3. Save the output of the above step, artificial data, in the new feature and target training variables.

[ ]:
```
# Write the code to apply oversample the data.

# Import the 'SMOTE' module from the 'imblearn.over_sampling' library.


# Call the 'SMOTE()' function and store it in the a variable.

# Call the 'fit_sample()' function.
```

Print the number of rows and columns in the original and artificial feature and target data:

[ ]:
```
# Print the number of rows and columns in the original and  resampled data.
```

Q: How many rows and columns are there in the original features data?

A:

Q: How many rows and columns are there in the artificially generated features data?

A:

Print the number of occurrences of labels in the artificially generated target data:

```
[ ]: # Display the number of occurrences of each label in the artificially target␣
     ↪data.
```

Q. Are the number of occurrances equal for all the labels?

A

After this activity, the training feature and target data should have the synthetic data such that all the labels have equal occurrances and the data is balanced

---

Activity 5: Random Forest Classification - Model Training Implement Random Forest Classification using `sklearn` module in the following way:

1. Deploy the model by importing the `RandomForestClassifier` class and create an object of this class.
2. Call the `fit()` function using the Random Forest Classifier object and print score using the `score()` function using the object created.

```
[ ]: # Import the required modules from the 'sklearn.ensemble' and 'sklearn.metrics'␣
     ↪libraries

     # Train the Random Forest Classifier
```

Q: What is the accuracy score?

A:

Q: Is it 100%?

A:

After this activity, a Random Forest Classification model object should be trained for multiclass classification.

Activity 6: Random Forest Classification - Model Prediction and Evaluation (Testing Set) **1.** Predict the values for the testing set by calling the `predict()` function on the Random Forest Classifier object.

2. Print the unique labels predicted using Random Forest Regression on training features.

3. Print the distribution of the labels predicted in the predicted target series for the testing features.

```
[ ]: # Make predictions using Random Forest Classifier model object

     # Make predictions on the test dataset by using the 'predict()' function.
     # Check the count of records classified under each label

     # Print the unique labels in the predicted series for testing features

     # Print the distribution labels in the predicted series for testing features
```

Q: Are all the label values predicted for the testing features data?

A:

Q: Which labels are predicted and not predicted by the Random Forest Classification model?

A:

---

4. Display the confusion matrix for the testing set:

```
[ ]: # Print the confusion matrix for the actual and predicted data of the test set␣
     ↪(if required)
```

5. Visualize the confusion matrix using the heatmap:

```
[ ]: # Create the heatmap for the confusion matrix

     # import the module


     # Create a series of all unique labels in actual target data

     # Create a Pandas DataFrame object for the confusion matrix created above␣
     ↪labelled with the classes.

     # Create a heatmap for the confusion matrix.
```

Note: Use `fmt='d'` as one of the parameters in the `heatmap()` to display the values in plain notation.

Q. Explain the heat map output in short based on the number of positive and negative outcomes for each class?

A

---

6. Display the classification report for the test set:

```
[ ]: # Print the classification report for the actual and predicted data of the␣
     ↪testing set (if required)
```

Q Which classes were identified correctly without any misclassification?

A

Q Which class has the lowest f1-score?

A

After this activity, labels should be predicted for the target columns using the test features set and the Random Forest Classifier model should be evaluated for the same.

Activity 7: Logistic Regression - Model Training Implement Logistic Regression Classification using `sklearn` module in the following way:

1. Deploy the model by importing the `LogisticRegression` class and create an object of this class.
2. Call the `fit()` function on the Logistic Regression object and print the score using the `score()` function.

[ ]:
```
# Build a logistic regression model using the 'sklearn' module.


# 1. First, call the 'LogisticRegression' module and store it in 'lg_clg'␣
↪variable.

# 2. Call the 'fit()' function with 'X_train' and 'y_train' as inputs.

# 3. Call the 'score()' function with 'X_train' and 'y_train' as inputs to␣
↪check the accuracy score of the model.
```

Q: What is the accuracy score?

A:

Q: Is it 100%?

A:

After this activity, a Logistic Regression model object should be trained for multiclass classification.

Activity 8: Logistic Regression - Model Prediction and Evaluation (Testing Set) **1.** Predict the values for the test set by calling the `predict()` function on the Logistic Regression object.

2. Print the unique labels predicted using Logistic Regression on test features.

3. Print the distribution of the labels predicted in the predicted target series for the test features.

```
[ ]: # Predict the values of 'target' by the logistic regression model on the test␣
     ↪set.

     # Predict the target for the test features data

     # Convert the predicted array into series

     # Print the unique labels in the predicted series for test features

     # Print the distribution labels in the predicted series for test features
```

Q: Are all the labels predicted for the test features data?

A:

Q: What are labels predicted and not predicted using Logistic Regression model object?

A:

---

4. Display the confusion matrix for the test set:

```
[ ]: # Print the confusion matrix for the actual and predicted data of the test set␣
     ↪(if required)
```

5. Visualize the confusion matrix using the heatmap:

```
[ ]: # Create the heatmap for the confusion matrix generated by Logistic Regression␣
     ↪testing predictions

     # Create a Pandas DataFrame object for the confusion matrix created above␣
     ↪labelled with the classes.


     # Create a heatmap for the confusion matrix.
```

6. Display the classification report for the test set:

```
[ ]: # Print the classification report for the actual and predicted data of the␣
     ↪testing set (if required)
```

Q Which classes were identified correctly without any misclassification?

A

Q Which class has the lowest f1-score?

A

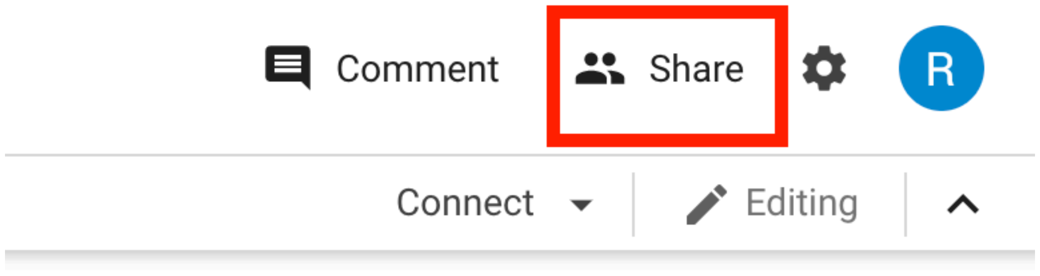
After this activity, labels should be predicted for the target columns using the test features set and the Logistic Regression model should be evaluated for the same.

Write your interpretation of the results here.

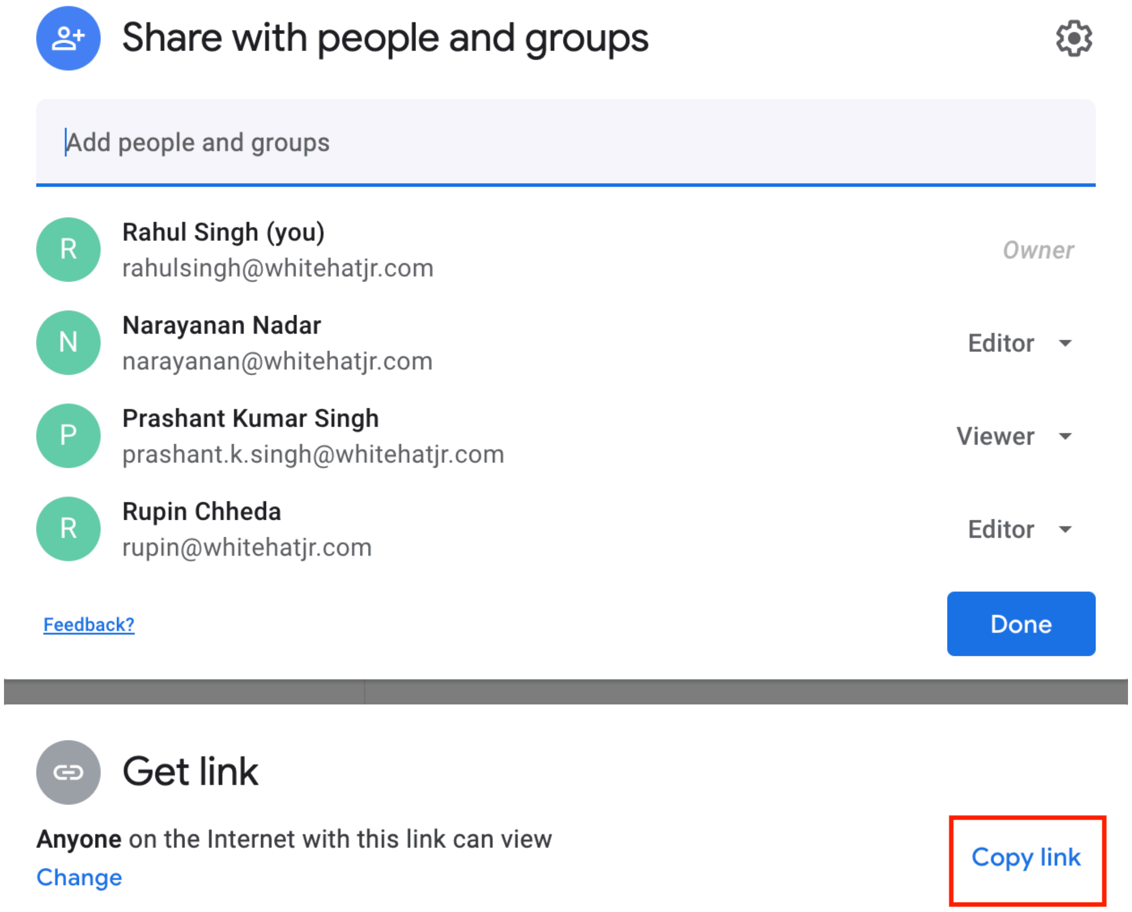
- Interpretation 1:
- Interpretation 2:

0.2.1 Submitting the Project:

1. After finishing the project, click on the **Share** button on the top right corner of the notebook. A new dialog box will appear.



2. In the dialog box, make sure that '**Anyone on the Internet with this link can view**' option is selected and then click on the **Copy link** button.


Share with people and groups

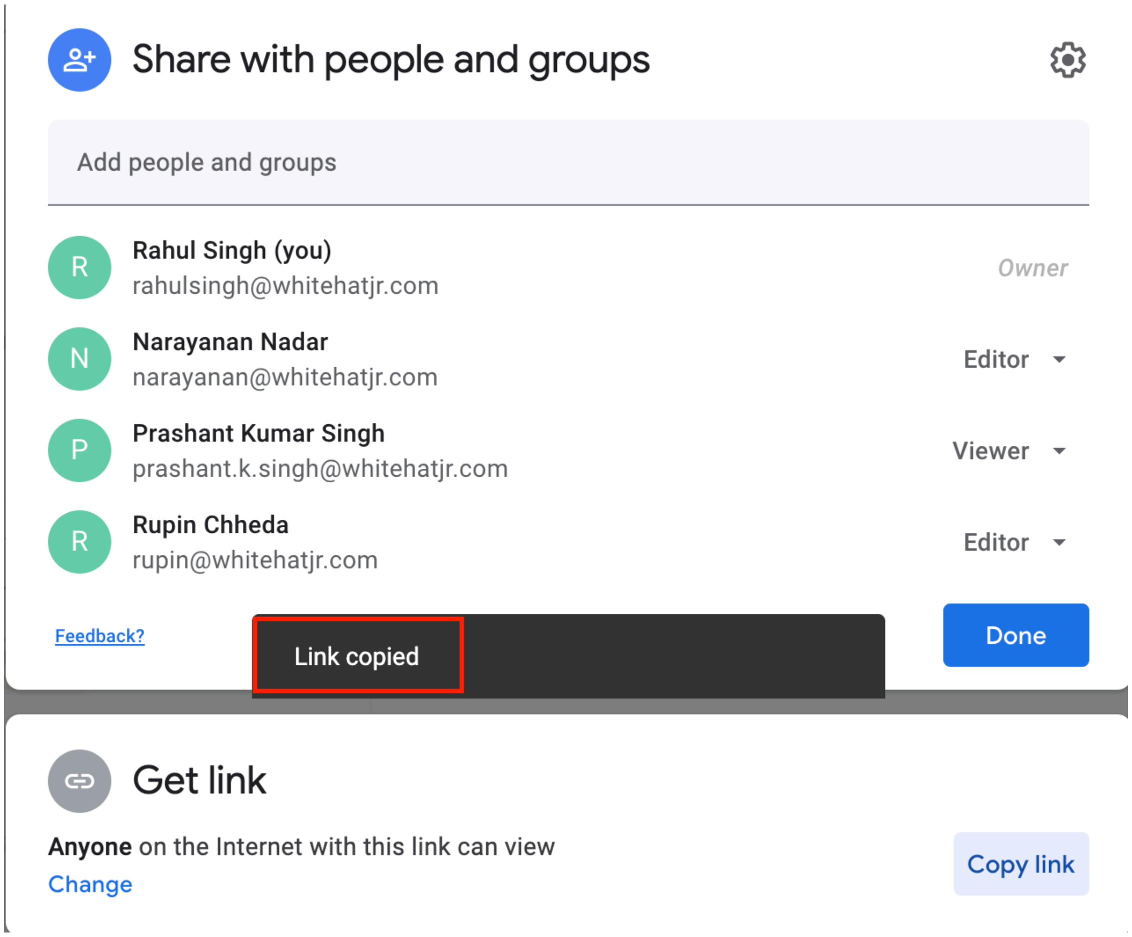
Add people and groups

Rahul Singh (you) rahulsingh@whitehatjr.com	Owner
Narayanan Nadar narayanan@whitehatjr.com	Editor
Prashant Kumar Singh prashant.k.singh@whitehatjr.com	Viewer
Rupin Chheda rupin@whitehatjr.com	Editor

Get link

Anyone on the Internet with this link can view

3. The link of the duplicate copy (named as **YYYY-MM-DD_StudentName_Project81**) of the notebook will get copied



Share with people and groups

Add people and groups

Rahul Singh (you) rahulsingh@whitehatjr.com — Owner

Narayanan Nadar narayanan@whitehatjr.com — Editor

Prashant Kumar Singh prashant.k.singh@whitehatjr.com — Viewer

Rupin Chheda rupin@whitehatjr.com — Editor

Feedback? Link copied Done

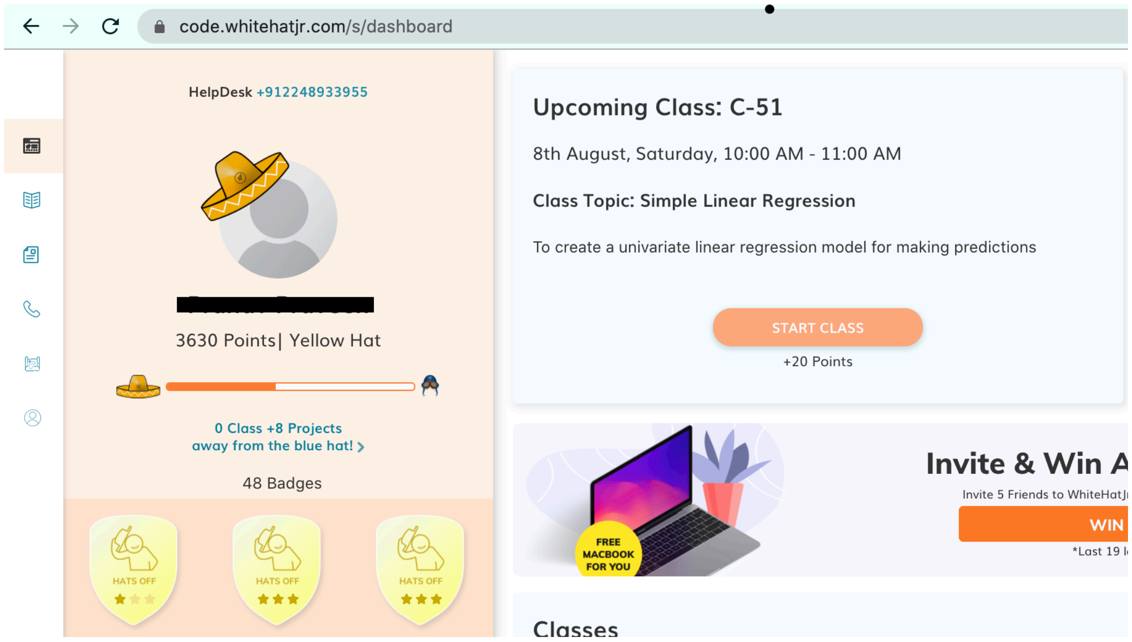
Get link

Anyone on the Internet with this link can view

Change

Copy link

4. Go to your dashboard and click on the **My Projects** option.



code.whitehatjr.com/s/dashboard

HelpDesk +912248933955

[redacted]

3630 Points| Yellow Hat

0 Class +8 Projects away from the blue hat!

48 Badges

Upcoming Class: C-51

8th August, Saturday, 10:00 AM - 11:00 AM

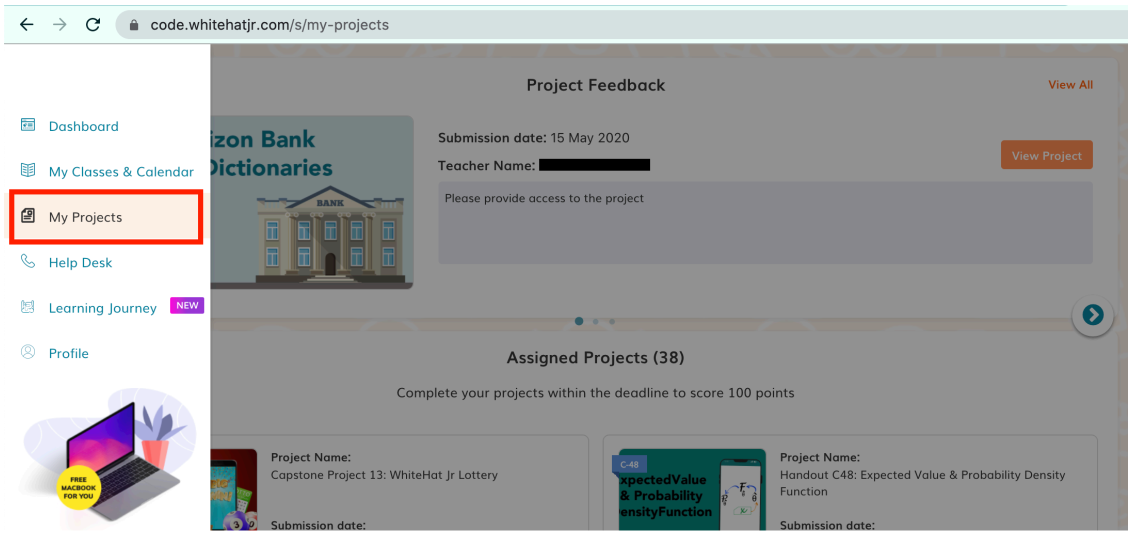
Class Topic: Simple Linear Regression

To create a univariate linear regression model for making predictions

START CLASS

+20 Points

Classes



Submission date: 15 May 2020

Teacher Name: [redacted]

Please provide access to the project

Complete your projects within the deadline to score 100 points

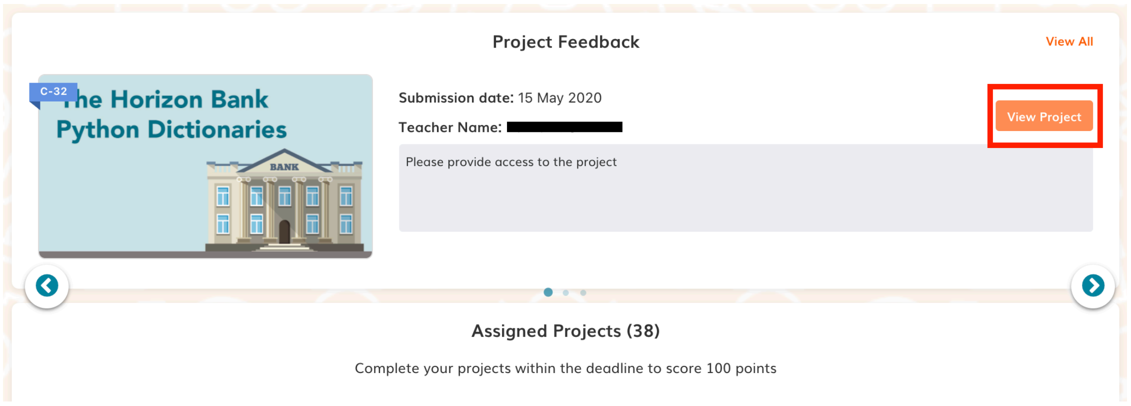
Project Name:
Capstone Project 13: WhiteHat Jr Lottery

Submission date:

Project Name:
Handout C48: Expected Value & Probability Density Function

Submission date:

5. Click on the **View Project** button for the project you want to submit.



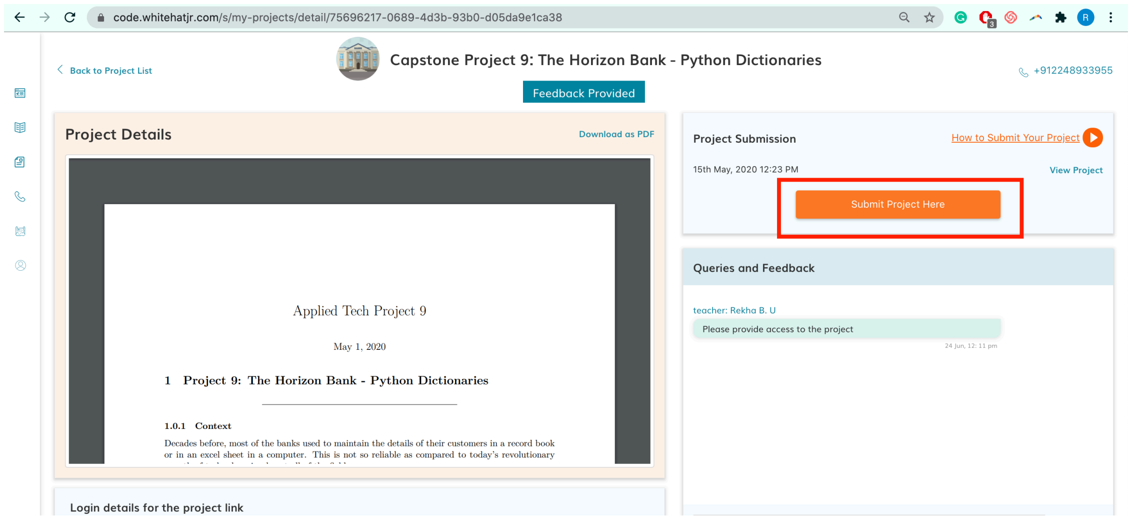
Submission date: 15 May 2020

Teacher Name: [redacted]

Please provide access to the project

Complete your projects within the deadline to score 100 points

6. Click on the **Submit Project Here** button.


code.whitehatjr.com/s/my-projects/detail/75696217-0689-4d3b-93b0-d05da9e1ca38
Back to Project List
Capstone Project 9: The Horizon Bank - Python Dictionaries
+912248933955
Feedback Provided
Project Details
Download as PDF
Project Submission
How to Submit Your Project
View Project
Submit Project Here
Queries and Feedback
Login details for the project link

15th May, 2020 12:23 PM

Applied Tech Project 9

May 1, 2020

1 Project 9: The Horizon Bank - Python Dictionaries

1.0.1 Context

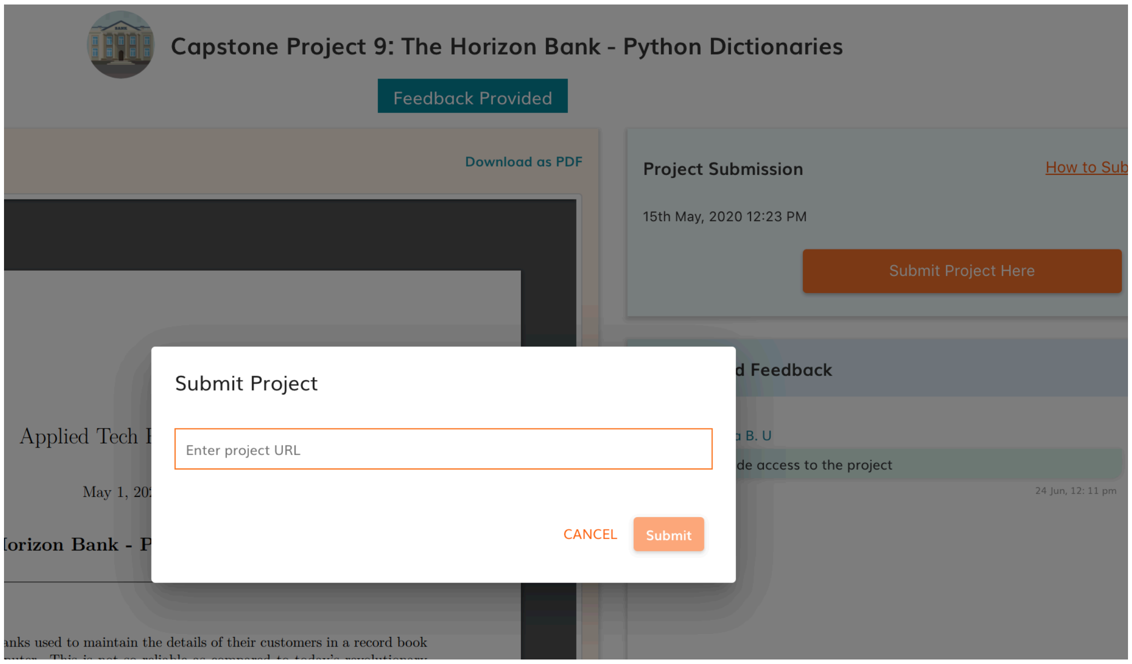
Decades before, most of the banks used to maintain the details of their customers in a record book or in an excel sheet in a computer. This is not so reliable as compared to today's revolutionary

teacher: Rekha B. U

Please provide access to the project

24 Jun, 12:11 pm

7. Paste the link to the project file named as **YYYY-MM-DD_StudentName_Project81** in the URL box and then click on the **Submit** button.


Capstone Project 9: The Horizon Bank - Python Dictionaries
Feedback Provided
Download as PDF
Project Submission
How to Sub
15th May, 2020 12:23 PM
Submit Project Here
Submit Project
Enter project URL
CANCEL
Submit