Applied Tech. Capstone Project 18

March 3, 2021

1 Capstone Project 18: Census Income Analysis

1.1 Instructions

1.1.1 Goal of the Project:

From class 67 to class 79, you learned the following concepts:

- Feature Encoding.
- Recursive Feature Elimination (RFE).
- Logistic Regression classification using `sklearn` module.

In this project, you will apply what you have learned in class 67 - 79 to achieve the following goals.

Main Goal: Create a Logistic Regression model classification model with ideal number of features selected using RFE.

1.1.2 Context

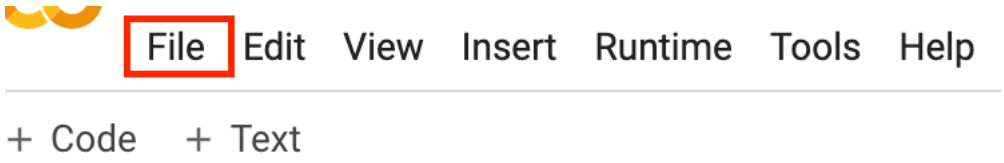
According to the government, census income is the income received by an individual regularly before payments for personal income taxes, medicare deductions, and so on. This information is asked annually from the people to record in the census. It helps to identify the eligible families for various funds and programs rolled out by communities and the government.

Getting Started Follow the steps described below to solve the project:

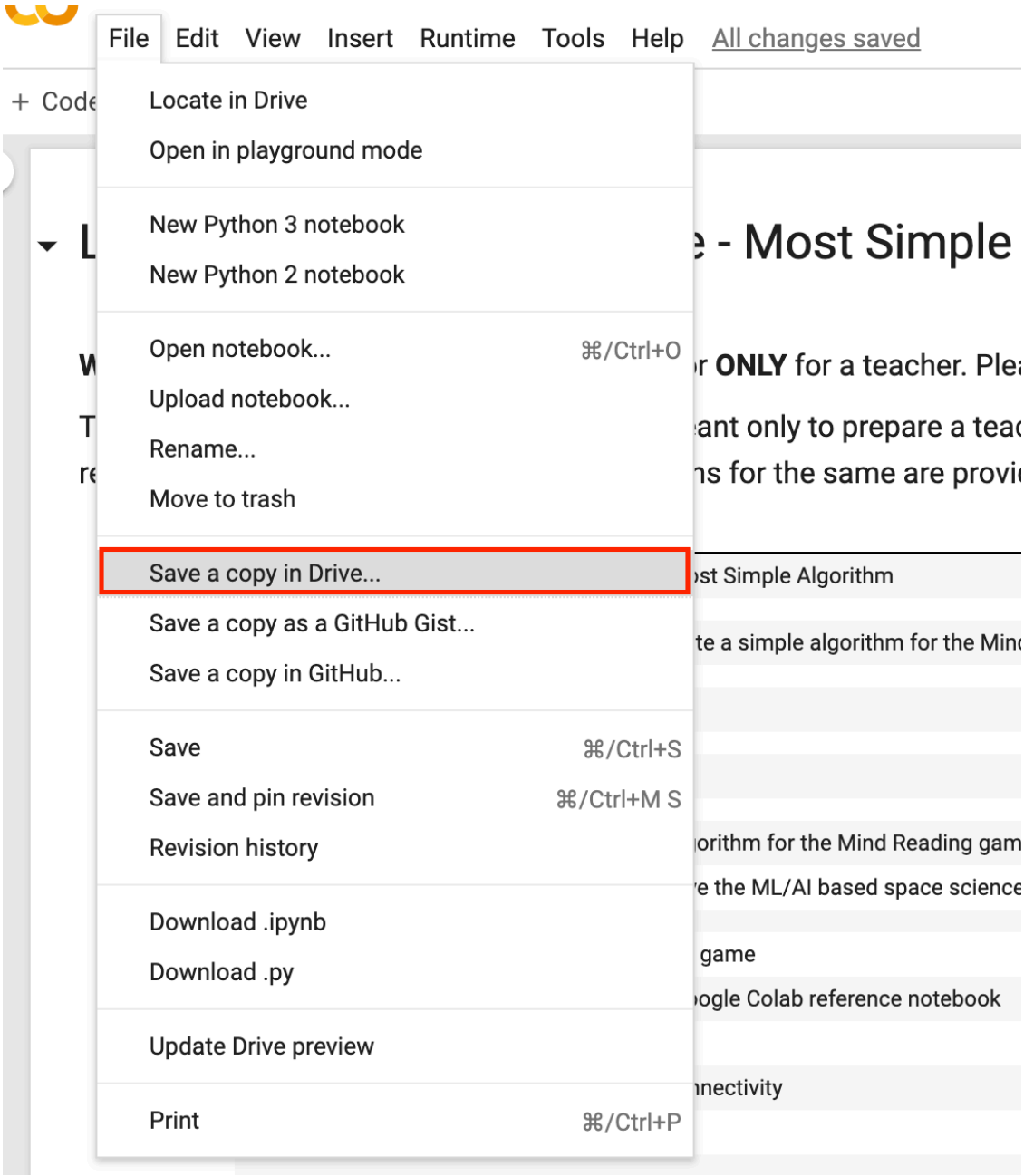
1. Click on the link provided below to open the Colab file for this project.

   https://colab.research.google.com/drive/1SJYyeTu9sl43k_kiozYNCndVely1e7Lr

2. Create the duplicate copy of the Colab file. Here are the steps to create the duplicate copy:
   - 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.



- After creating the duplicate copy of the notebook, please rename it in the **YYYY-MM-**

DD_StudentName_CapstoneProject18 format.

3. Now, write your code in the prescribed code cells.

1.1.3 Problem Statement

The dataset is extracted from 1994 Census Bureau. The data includes an instance of anonymous individual records with features like work-experience, age, gender, country, and so on. Also have divided the records into two labels with people having a salary **more than 50K or less than equal to 50K** so that they can determine the eligibility of individuals for government opted programs.

Looks like a very interesting dataset and as a data scientist, your job is to build a prediction model to predict whether a particular individual has an annual income of **<=50k** or **>50k**.

Things To Do:

1. Importing and Analysing the Dataset
2. Data Cleaning
3. Feature Engineering
4. Train-Test Split
5. Data Standardisation
6. Logistic Regression - Model Training
7. Model Prediction and Evaluation
8. Features Selection Using RFE
9. Model Training and Prediction Using Ideal Features

1.1.4 Dataset Description

The dataset includes 32561 instances with 14 features and 1 target column which can be briefed as:

- `age`: age of the person, Integer
- `work-class`: employment information about the individual, Categorical
- `fnlwgt`: unknown weights, Integer
- `education`: highest level of education obtained, Categorical
- `education-years`: number of years of education, Integer
- `marital-status`: marital status of the person, Categorical
- `occupation`: job title, Categorical
- `relationship`: individual relation in the family-like wife, husband, and so on. Categorical
- `race`: Categorical
- `sex`: gender, Male or Female
- `capital-gain`: gain from sources other than salary/wages, Integer
- `capital-loss`: loss from sources other than salary/wages, Integer
- `hours-per-week`: hours worked per week, Integer

- `native-country`: name of the native country, Categorical
- `income-group`: annual income, Categorical, **<=50k** or **>50k**

Notes: 1. The dataset has no header row for the column name. (Can add column names manually) 2. There are invalid values in the dataset marked as **"?"**. 3. As the information about **fnlwgt** is non-existent it can be removed before model training. 4. Take note of the **whitespaces (" ")** throughout the dataset.

Dataset Credits: https://archive.ics.uci.edu/ml/datasets/adult

Dataset Creator:

```
Ronny Kohavi and Barry Becker
Data Mining and Visualization
Silicon Graphics.
e-mail: ronnyk '@' live.com for questions.
```

Activity 1: Importing and Analysing the Dataset In this activity, we have to load the dataset and analyse it.

Perform the following tasks: - Load the dataset into a DataFrame. - Rename the columns with the given list. - Verify the number of rows and columns. - Print the information of the DataFrame.

1. Start with importing all the required modules:

[ ]:
```
# Import modules
```

2. Create a Pandas DataFrame for the **Adult Income** dataset using the below link with `header=None`. > **Dataset Link:** https://student-datasets-bucket.s3.ap-south-1.amazonaws.com/whitehat-ds-datasets/adult.csv

3. Print the first five rows of the dataset:

[ ]:
```
# Load the Adult Income dataset into DataFrame.
```

4. Rename the columns by applying the `rename()` function using the following column list:

```
column_name =['age', 'workclass', 'fnlwgt', 'education', 'education-years', 'marital-statu
```

[ ]:
```
# Rename the column names in the DataFrame using the list given above.

# Create the list


# Rename the columns using 'rename()'

# Print the first five rows of the DataFrame
```

Hint:

Syntax for `rename()` function:

`DataFrame.rename(columns={old_column_name:new_column_name})`

5. Verify the number of rows and columns in the DataFrame:

```
[ ]: # Print the number of rows and columns of the DataFrame
```

6. Get the information of the DataFrame:

```
[ ]: # Get the information of the DataFrame
```

Q: Which is the target column?

A:

7. Print the labels in the target column and their distribution as well:

```
[ ]: # Check the distribution of the labels in the target column.
```

Q: Which target label has more records?

A:

After performing this activity, you must obtain the DataFrame with renamed columns and the target column identified.

Activity 2: Data Cleaning In this activity, we need to clean the DataFrame step by step.

Perform the following tasks: - Check for the null or missing values in the DataFrame. - Observe the categories in column `native-country`, `workclass`, and `occupation`. - Replace the invalid `" ?"` values in the columns with `np.nan` using `replace()` function. - Drop the rows having `nan` values using the `dropna()` function.

1. Verify the missing values in the DataFrame:

```
[ ]: # Check for null values in the DataFrame.
```

Q: Are there any missing/null values that can be observed in the DataFrame?

A:

2. Observe the unique categories in columns `native-country`, `workclass`, and `occupation` to find the invalid values:

```
[ ]: # Print the distribution of the columns mentioned to find the invalid values.

     # Print the categories in column 'native-country'

     # Print the categories in column 'workclass'

     # Print the categories in column 'occupation'
```

Q: Is there any invalid value or category in any of the three columns?

A:

---

3. Replace the invalid values with `np.nan` and verify the number of null values in the DataFrame again.

```
[ ]: # Replace the invalid values ' ?' with 'np.nan'.


     # Check for null values in the DataFrame again.
```

Q: Are there any missing/null values that can be observed in the DataFrame?

A:

---

4. Delete the rows having invalid values and drop the column `fnlwgt`. Print the number of rows of the DataFrame after dropping invalid values:

```
[ ]: # Delete the rows with invalid values and the column not required

     # Delete the rows with the 'dropna()' function

     # Delete the column with the 'drop()' function


     # Print the number of rows and columns in the DataFrame.
```

After this activity, the DataFrame should neither have any null or invalid values nor the `fnlwgt` column.

Activity 3: Feature Engineering The dataset contains certain features that are categorical. To convert these features into numerical ones, use the `map()` and `get_dummies()` function.

Perform the following tasks for feature engineering:

- Create a list of numerical columns.
- Map the values of the column `gender` to:
  - `Male: 0`
  - `Female: 1`
- Map the values of the column `income-group` to:
  - `<=50K: 0`
  - `>50K: 1`
- Create a list of categorical columns.
- Perform **one-hot encoding** to obtain numeric values for the rest of the categorical columns.

---

1. Separate the numeric columns first for that create a list of numeric columns using `select_dtypes()` function:

```
[ ]: # Create a list of numeric columns names using 'select_dtypes()'.
```

2. Map the labels of the column `gender` to convert it into a numerical attribute using the `map()` function: - `Male` to `0` - `Female` to `1`

```
[ ]: # Map the 'sex' column and verify the distribution of labels.

     # Print the distribution before mapping


     # Map the values of the column to convert the categorical values to integer

     # Print the distribution after mapping
```

3. Map the labels of the column `income-group` to convert it into a numerical attribute from categorical one using `map()` function: - `<=50K` to `0` - `>50K` to `1`

```
[ ]: # Map the 'income-group' column and verify the distribution of labels.

     # Print the distribution before mapping


     # Map the values of the column to convert the categorical values to integer

     # Print the distribution after mapping
```

4. Create a list of categorical columns names using `select_dtypes()` function:

```
[ ]: # Create the list of categorical columns names using 'select_dtypes()'.
```

5. Perform **one-hot encoding** on the columns of the DataFrame in the list above and save it in a **dummy DataFrame**. Also use parameter `drop_first= True` in the `get_dummies()` function.

Recall: *This process of obtaining numeric values from non-numeric categorical values is called* ***one-hot encoding****. In this process a column is added for each of the categories in a particular feature and value in the columns will be binary* `0` *and* `1` *based on the original value in the feature and the category column. The* `get_dummies()` *function can be used to apply* ***one-hot encoding*** *to the non-numeric categorical feature columns.*

```
[ ]: # Create a 'income_dummies_df' DataFrame using the 'get_dummies()' function on␣
      ↪the non-numeric categorical columns
```

6. Drop the non-numeric categorical columns from the original income DataFrame using the `drop()` function:

```
[ ]: # Drop the categorical columns from the Income DataFrame `income_df`
```

7. Concat the income DataFrame and the dummy DataFrame to create the final DataFrame for the model.

Print the first five values of the final DataFrame:

```
[ ]: # Concat the income DataFrame and dummy DataFrame using 'concat()' function
```

8. Get the information of the DataFrame to verify the final columns and their data types:

```
[ ]: # Get the information of the DataFrame
```

Q: How many columns are present in the final DataFrame?

A:

Q: What is the data type of the columns in the final DataFrame?

A:

After this activity, the DataFrame should not have any non-numeric columns.

Activity 4: Train-Test Split We need to predict the value of the `income-group` variable, using other variables. Thus, `income-group` is the target or dependent variable and other columns except `income-group` are the features or 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`:

```
[ ]: # Split the training and testing data

     # Import the module
```

After this activity, the feature and target data should be distributed in training and testing data

---

Activity 5: Data Standardisation To avoid `ConvergenceWarning` message - That is to scale the data using one of the normalisation methods, for instance, standard normalisation.

1. Create a function `standard_scalar()` to normalise the numeric columns of `X_train` and `X_test` data-frames using the standard normalisation method:

```
[ ]: # Normalise the train and test data-frames using the standard normalisation␣
      ↪method.

     # Define the 'standard_scalar()' function for calculating Z-scores
```

```
# Create the DataFrames norm_X_train and norm_X_train

# Apply the 'standard_scalar()' on X_train on numeric columns using apply()␣
↪function and get the descriptive statistics of the normalised X_train

# Apply the 'standard_scalar()' on X_test on numeric columns using apply()␣
↪function and get the descriptive statistics of the normalised X_test
```

After this activity, training and testing feature data should be normalised using Data Standardisation.

Activity 6: Logistic Regression - Model Training Implement Logistic Regression Classification using `sklearn` module to estimate the values of β coefficients 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.

[ ]:
```
# Deploy the 'LogisticRegression' model using the 'fit()' function.
```

After this activity, a multi-variate logistic regression should be trained with all features.

Activity 7: Model Prediction and Evaluation **1.** Predict the values for both training and test sets by calling the `predict()` function on the Logistic Regression object:

[ ]:
```
# Make predictions on the test dataset by using the 'predict()' function.
```

2. Display the confusion matrix:

[ ]:
```
# Display the results of confusion_matrix
```

Q: What is the positive outcome out of both labels?

A:

Q: Write the count of True Positives and True Negatives?

A:

3. Print the classification report values to evaluate the accuracy of your model:

[ ]:
```
# Display the results of classification_report
```

Q Write the f1-score of both labels?

A:

After this activity, a multi-variate logistic regression model is used to predict and evaluate using all the features.

Activity 8: Features Selection Using RFE Select the relevant features from all the features that contribute the most to classifying individuals in income-groups using RFE.

Steps:

1. Create an empty dictionary and store it in a variable.

2. Create a `for` loop that iterates through all the columns in the normalised training data-frame. Inside the loop:

- Create an object of the Logistic Regression class and store it in a variable.
- Create an object of RFE class and store it in a variable. Inside the RFE class constructor, pass the object of logistic regression and the number of features to be selected by RFE as inputs.
- Train the model using the `fit()` function of the `RFE` class to train a logistic regression model on the train set with `i` number of features where `i` goes from `1` to the number of columns in the training dataset.
- Create a list to store the important features using the `support_` attribute.
- Create a new data-frame having the features selected by RFE store in a variable.
- Create another Logistic Regression object, store it in a variable and build a logistic regression model using the new training DataFrame created using the rfe features data-frame and the target series.
- Predict the target values for the normalised test set (containing the feature(s) selected by RFE) by calling the `predict()` function on the recent model object.
- Calculate f1-scores using the function `f1_score()` function of `sklearn.metrics` module that returns a NumPy array containing f1-scores for both the classes. Store the array in a variable called `f1_scores_array`.

```
The sytax for the `f1_score()` is given as:

  >**Syntax:** `f1_score(y_true, y_pred, average = None)`

    Where,

    **a.** `y_true`:  the actual labels

    **b.** `y_pred`: the predicted labels

    **c.** `average = None`: parameter returns the scores for each class.
```

- Add the number of selected features and the corresponding features & f1-scores as key-value pairs in the dictionary.

Note:
As the number of features is very high, the code will be a computationally heavy program. It will require very GPU to process the code faster. It will take some time to learn the feature variables through the training data and then make predictions on the test data.

To turn on the **GPU** in google colab follow the steps below: 1. Click on the **Edit** menu option on the top-left. 2. Click on the **Notebook settings** option from the menu. A pop-up will appear. 3. Click on the drop-down for selecting **Hardware accelerator**. 4. Select **GPU** from the drop-down options. 5. Click on **Save**.

[ ]:
```
# Create a dictionary containing the different combination of features selected␣
↪by RFE and their corresponding f1-scores.

# Import the libraries


# Create the empty dictionary.

# Create a 'for' loop.

  # Create the Logistic Regression Model

  # Create the RFE model with 'i' number of features

  # Train the rfe model on the normalised training data using 'fit()'

  # Create a list of important features chosen by RFE.

  # Create the normalised training DataFrame with rfe features

  # Create the logistic regression

  # Train the model normalised training DataFrame with rfe features using␣
↪'fit()'

  # Predict 'y' values only for the test set as generally, they are predicted␣
↪quite accurately for the train set.

  # Calculate the f1-score

  # Add the name of features and f1-scores in the dictionary
```

6. Print the dictionary with features and f1-scores.

[ ]:
```
# Print the dictionary
```

7. Convert the dictionary into a DataFrame by using the`from_dict()` function of the DataFrame.

Note Set the `pd.options.display.max_colwidth` to `200`

[ ]:
```
# Convert the dictionary to the DataFrame
```

Q: How many features are required for the best f1-scores and why?

A:

After this activity, rfe is used to find the ideal features for logistic regression

---

Activity 9: Model Training and Prediction Using Ideal Features **1.** Create the logistic regression model again using RFE with the ideal number of features and predict the target variable:

[ ]:
```
# Logistic Regression with the ideal number of features and predict the target.

# Create the Logistic Regression Model


# Create the RFE model with ideal number of features

# Train the rfe model on the normalised training data

# Create a list of important features chosen by RFE.

# Create the normalised training DataFrame with rfe features

# Create the Regression Model again

# Train the model with the normalised training features DataFrame with best rfe␣
↪features and target training DataFrame

# Predict the target using the normalised test DataFrame with rfe features

# Calculate the final f1-score and print it
```

Hint: Create the model using the same steps mentioned in the **Features Selection Activity.**

Q: What is the final f1-score?

A:

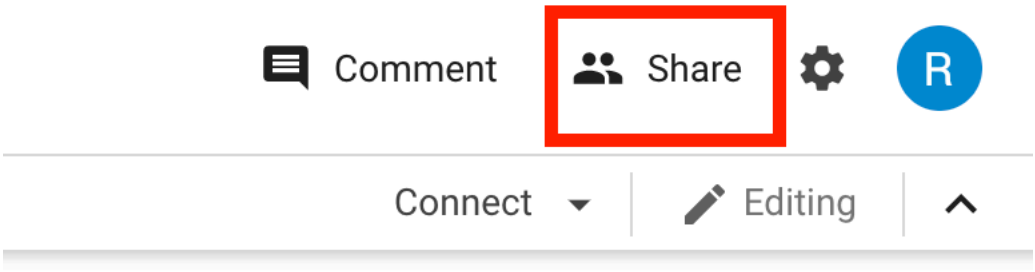
Write your interpretation of the results here.

- Interpretation 1:
- Interpretation 2:
- Interpretation 3:

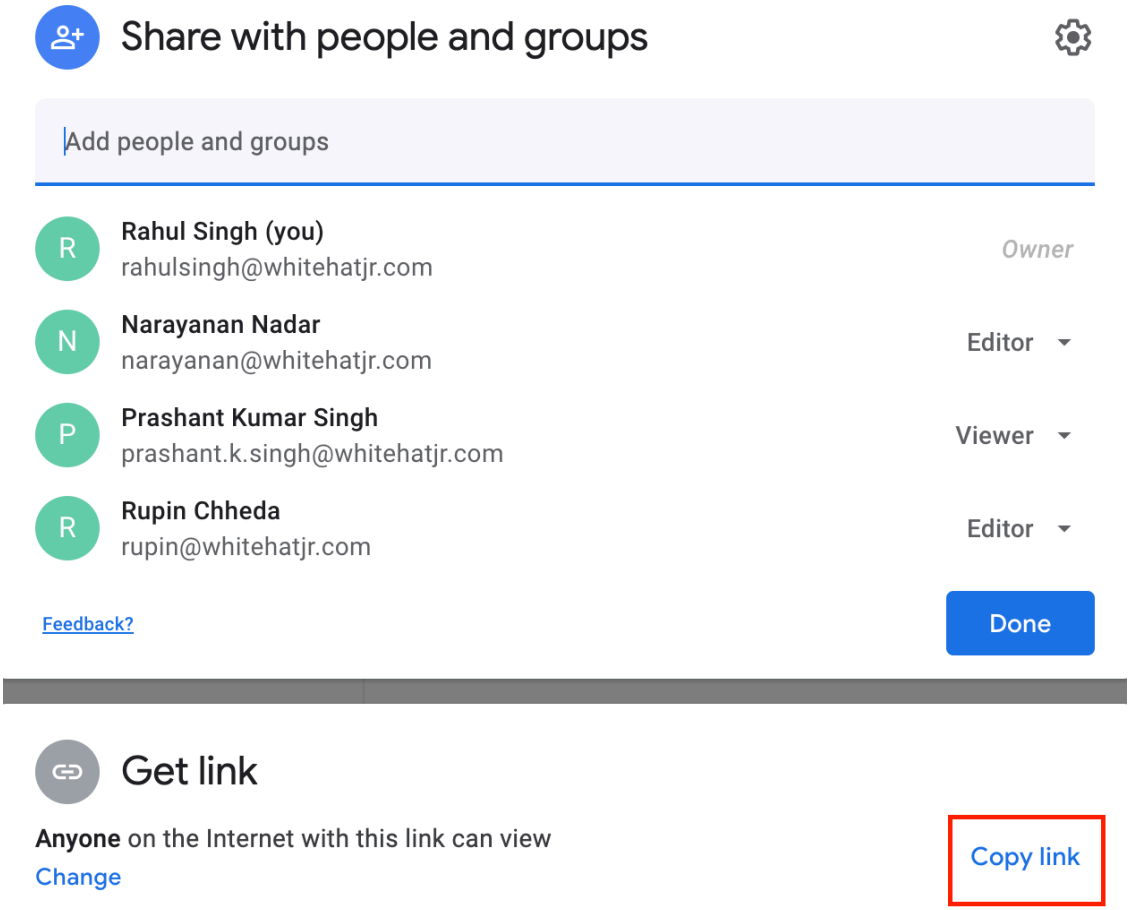
After this activity, a Logistic Regression model should be ready with ideal number of features to accurately predict the income group of the people that is to predict whether an individual has annual income `less or equal than 50K (label 0)` or `more than 50K (label 1)` based on the features selected.

1.1.5 Submitting the Project:

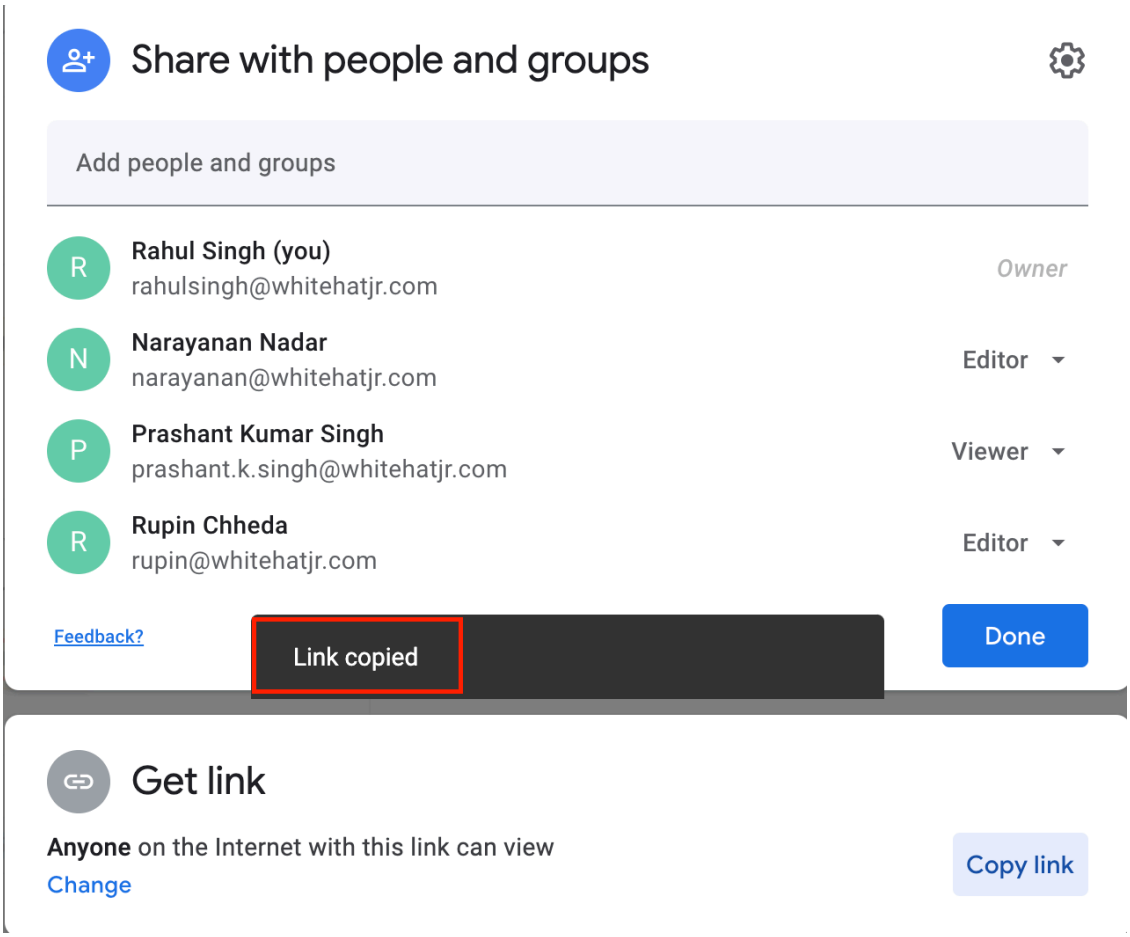
1. After finishing the project, click on the **Share** button on the top right corner of the notebook. A new dialog box will appear.



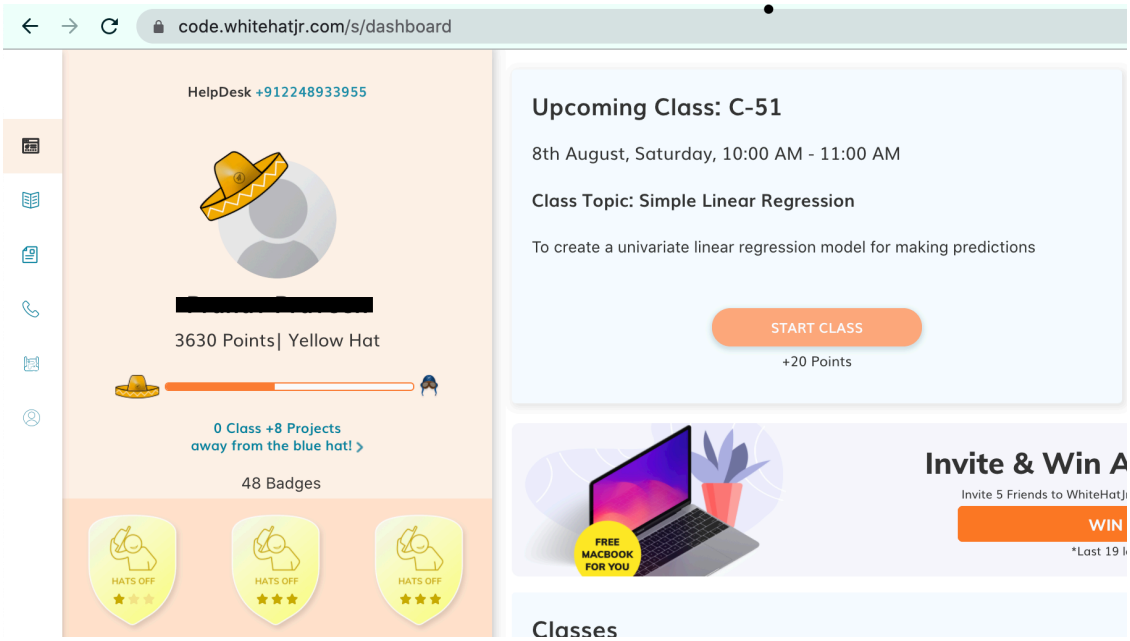
2. In the dialog box, make sure that '**Anyone on the Internet with this link can view**' option is selected and then click on the **Copy link** button.

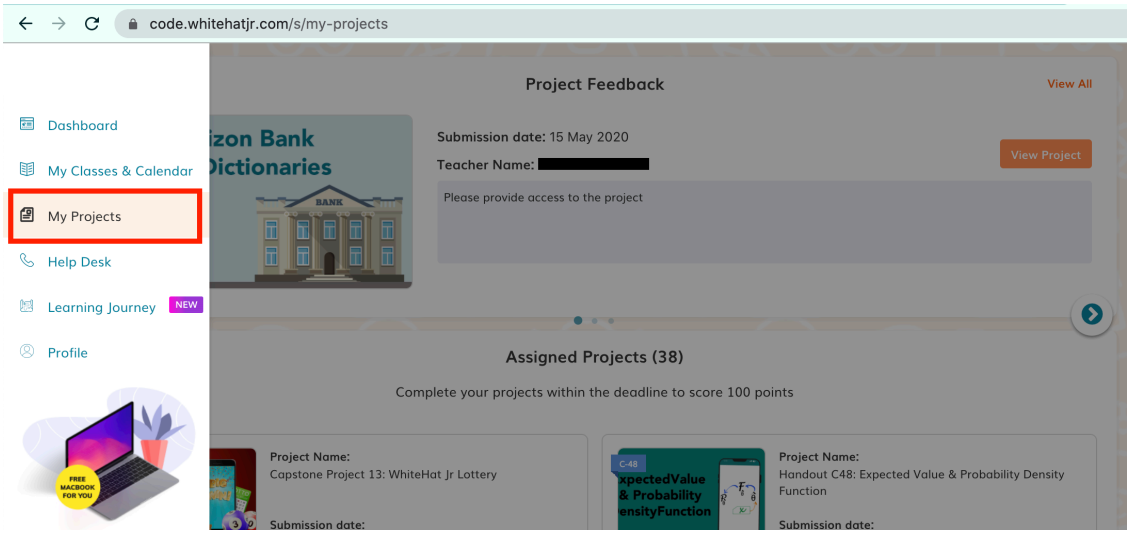


3. The link of the duplicate copy (named as **YYYY-MM-DD_StudentName_CapstoneProject18**) of the notebook will get copied

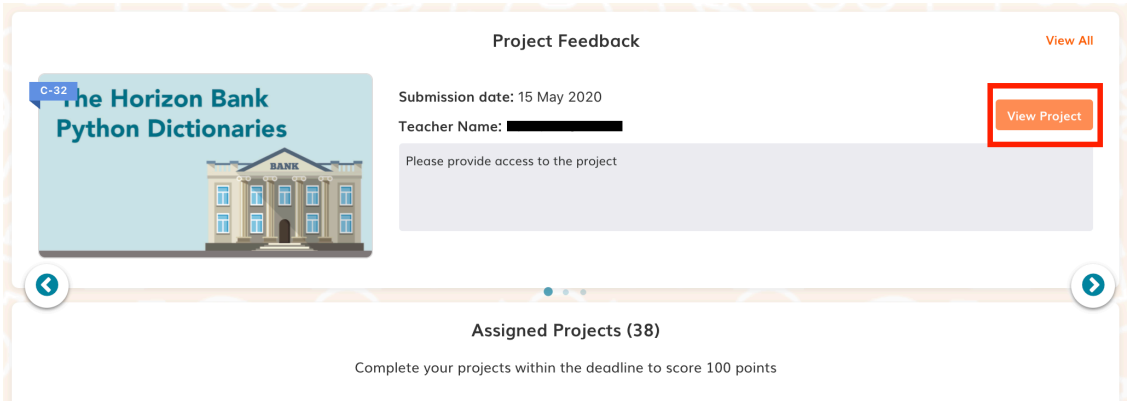


4. Go to your dashboard and click on the **My Projects** option.

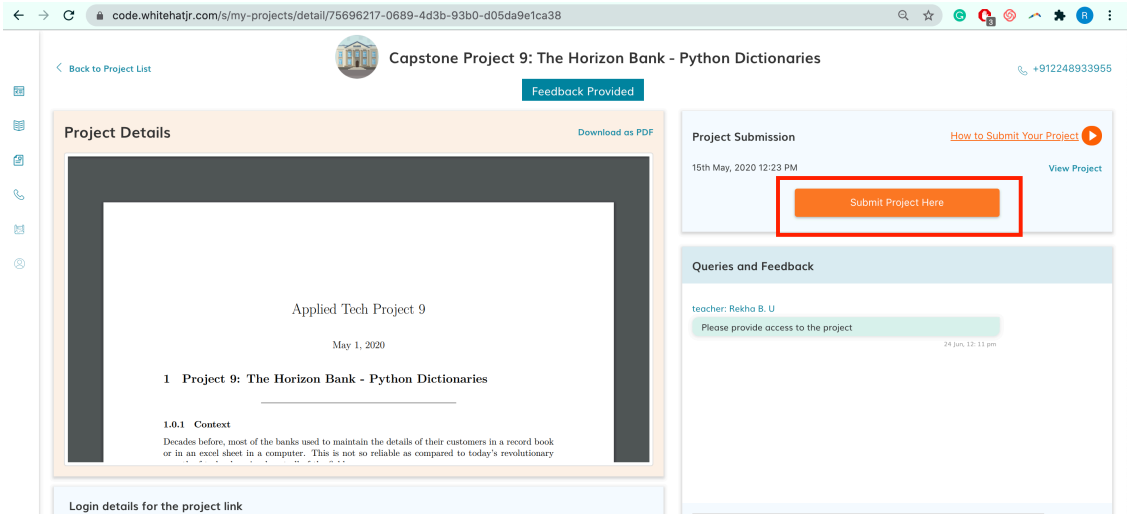




5. Click on the **View Project** button for the project you want to submit.



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



7. Paste the link to the project file named as **YYYY-MM-DD_StudentName_CapstoneProject18** in the URL box and then click on the **Submit** button.