**User Manual for Data Import and Analysis**

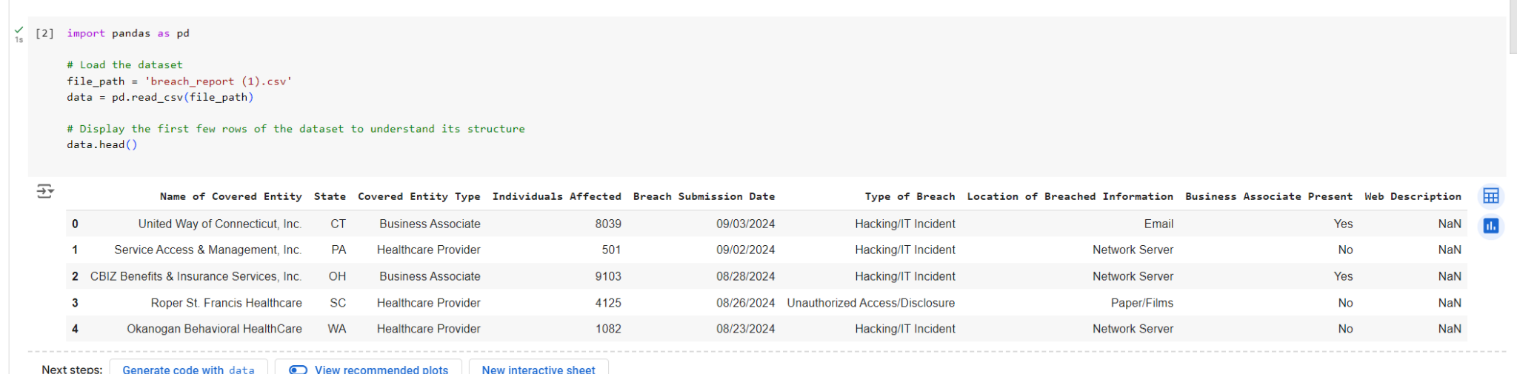
This guide walks through the process of importing the healthcare breach dataset and performing analysis using Python, specifically with libraries like Pandas, Matplotlib, and others.

**1. Importing the Dataset**

In this section, we demonstrate how to import the dataset into Python using the pandas library.

Steps:

* First, ensure that you have the Pandas library installed (pip install pandas).
* Import the dataset by specifying the file path and using the pd.read\_csv() function.



**Explanation:**

* pd.read\_csv() is used to read the CSV file containing the breach data.
* The data.head() function prints the first five rows of the dataset, giving you an initial overview of the dataset’s structure, including the columns like "Name of Covered Entity," "State," "Individuals Affected," etc.

**2. Frequency Analysis and Visualization**

This section demonstrates how to perform frequency analysis and create visualizations using the matplotlib library.

**Steps:**

* Use the value\_counts() method to count occurrences of specific variables, such as types of breaches, states, and entity types.
* Plot bar charts using Matplotlib to visualize the results.



**Explanation:**

* The value\_counts() method counts the occurrences of different breach types, states, and entity types.
* Bar charts are created to visually represent the distribution of different variables like "Type of Breach" and "State." The plot(kind='bar') generates the bar chart, and the labels and titles help contextualize the data for easy understanding.

**3. Time Series and Descriptive Analysis**

In this section, we analyze the breach submission dates to understand trends over time and perform a descriptive analysis of individuals affected.

**Steps:**

* Convert the breach submission date to datetime format using Pandas.
* Extract 'Year' and 'Month' for time-based analysis and plot the number of breaches over time.
* Perform a descriptive analysis and plot the distribution of individuals affected.



**Explanation:**

* We first convert the "Breach Submission Date" to a proper date format using pd.to\_datetime().
* We extract the year and month from the submission date for time series analysis.
* A line chart is plotted to visualize how the number of breaches changes over time, helping us detect any patterns or trends.
* Additionally, we conducted a descriptive analysis of the "Individuals Affected" to understand the impact of breaches.

**4. Logistic Regression Model Setup and Training**

In this section, we set up and train a **Logistic Regression** model.

**Steps:**

**Import Required Libraries:** We import the necessary libraries for data processing, model creation, and evaluation.

 **Define Features and Target:** We define the features that will be used for the prediction and the target variable (breach type).

 **Preprocessing Pipeline:** We set up a pipeline to handle categorical data encoding and missing values.

 **Logistic Regression Pipeline:** We set up a pipeline that combines preprocessing and Logistic Regression.

 **Train-Test Split:** The data is split into training and test sets to evaluate model performance.

 **Model Training and Evaluation:** We train the Logistic Regression model and evaluate its performance.

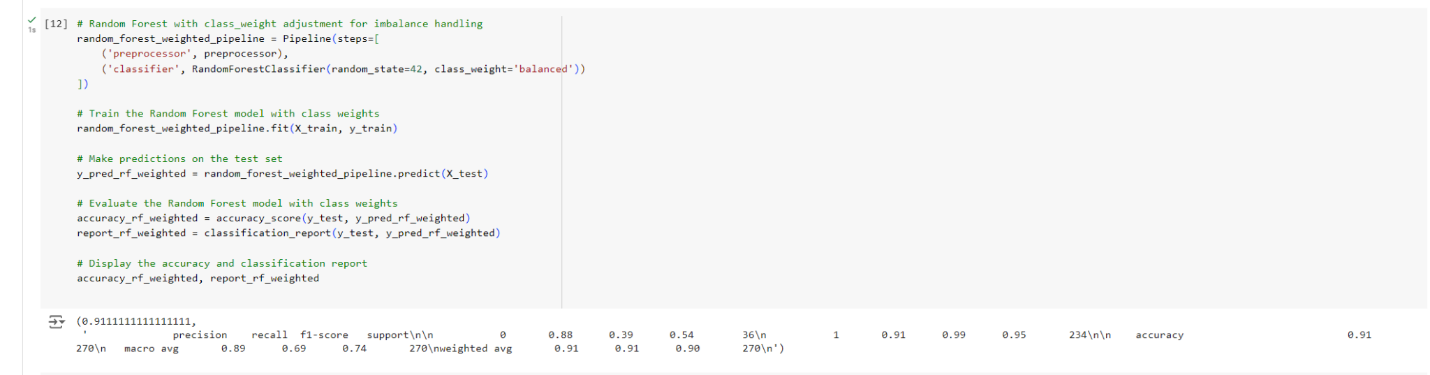


**2. Random Forest Model with Class Weight Adjustment (Screenshot 2)**

In this section, we train a **Random Forest** model, adjusting for class imbalance using the class\_weight parameter.

**Steps:**

1. **Random Forest with Class Weight Adjustment:** The class\_weight parameter is set to 'balanced' to handle class imbalance issues.
2. **Model Training and Prediction:** The Random Forest model is trained and used to make predictions on the test set.
3. **Model Evaluation:** The accuracy and classification report are generated to assess the model's performance.

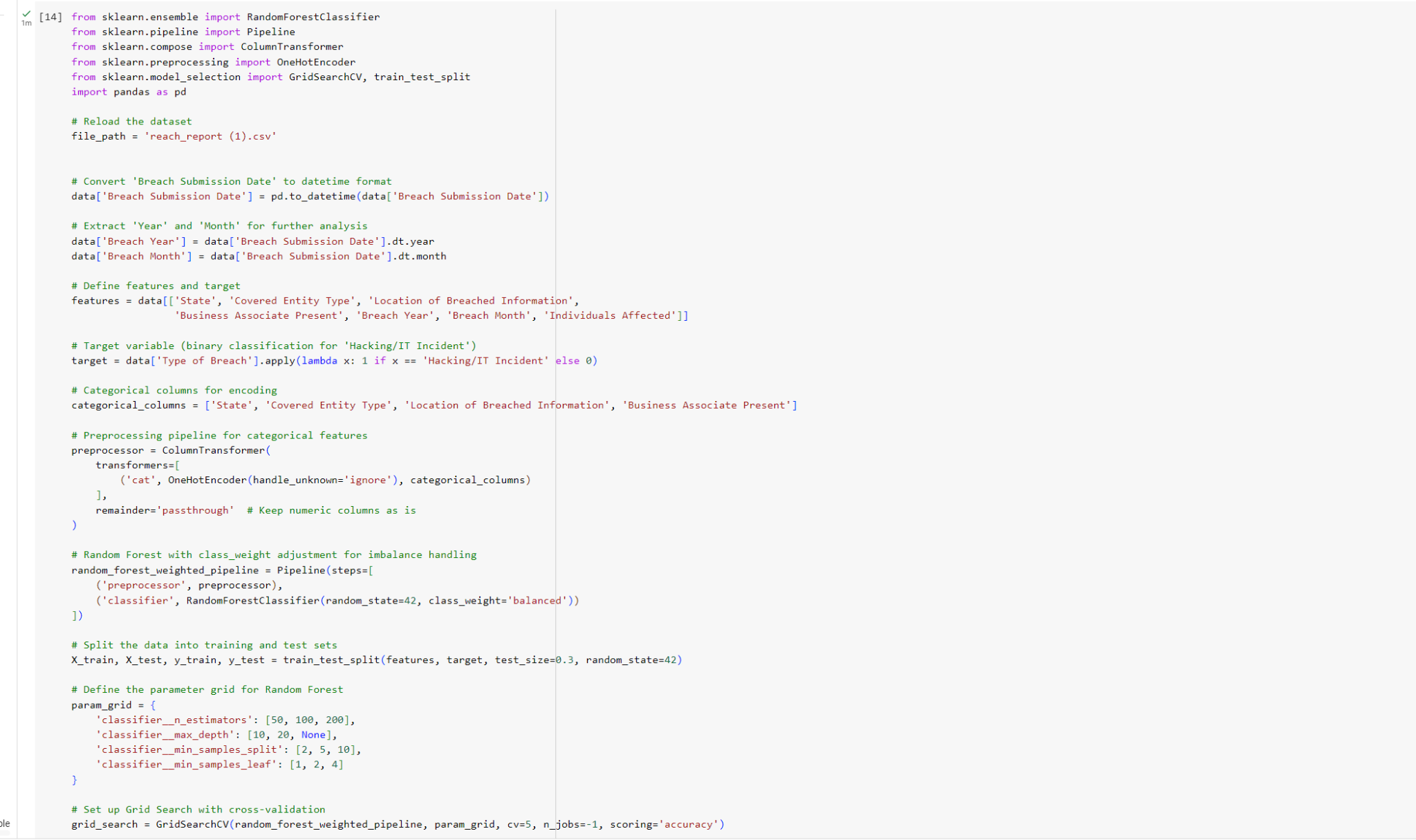


**3. Hyperparameter Tuning with GridSearch for Random Forest (Screenshot 3)**

In this section, we perform **hyperparameter tuning** for the Random Forest model using GridSearchCV to improve its performance.

**Steps:**

1. **Set Up the Parameter Grid:** Define a parameter grid for tuning the Random Forest model’s hyperparameters.
2. **Grid Search with Cross-Validation:** We set up GridSearchCV to explore the hyperparameter combinations and find the best set of parameters.
3. **Train the Tuned Model:** The model is trained using the best-found parameters.
4. **Evaluate the Tuned Mo**
5. **del:** Evaluate the model's performance using the test set with the tuned parameters.



**4. Model Setup and Training**

This section explains how to set up and train three different machine learning models: **Logistic Regression**, **Decision Tree**, and **Neural Network**.

**Steps:**

1. **Import Required Libraries:** Begin by importing the necessary libraries for data processing, model creation, and evaluation.
2. **Convert Date Columns:** Convert the "Breach Submission Date" to datetime format and extract "Year" and "Month" for time-based analysis.
3. **Define Features and Target:** Select the relevant features and set the target variable to predict hacking incidents as class 1.
4. **Preprocessing Pipeline:** Define a preprocessing pipeline to encode categorical variables.
5. **Model Pipelines:** Set up pipelines for Logistic Regression, Decision Tree, and Neural Network models.
6. **Train and Evaluate Models:** Train each model on the training data and evaluate the performance using accuracy and classification reports.

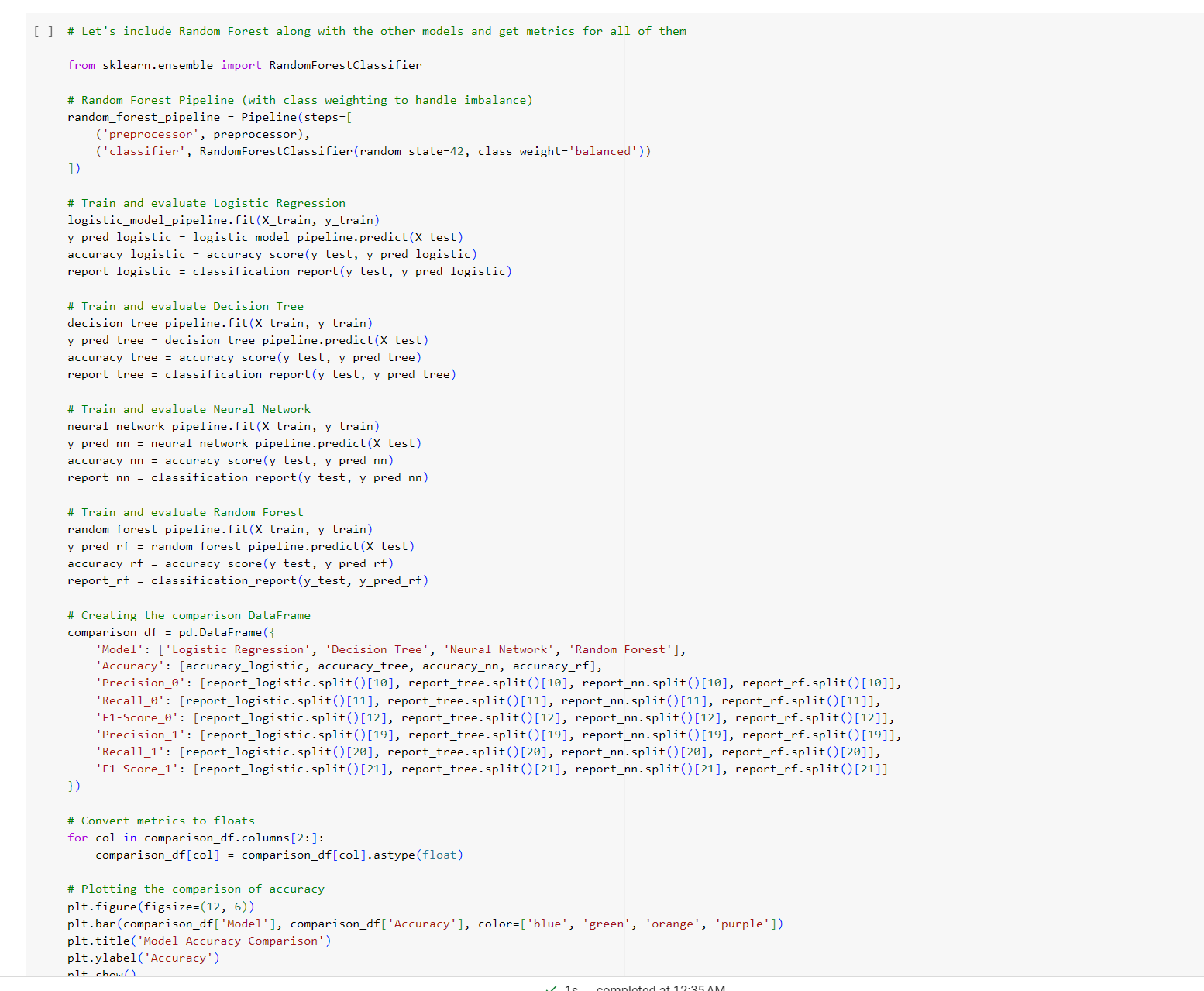


**5. Random Forest Model and Model Comparison**

This section shows how to include a **Random Forest** model along with the other models and compare their metrics.

**Steps:**

1. **Random Forest Pipeline:** Define a pipeline for the Random Forest model, with class weighting to handle imbalance.
2. **Train and Evaluate Random Forest:** Train the Random Forest model and evaluate it on the test set.
3. **Generate Comparison Table:** Collect accuracy and precision, recall, and F1-score for each model and store them in a comparison table.
4. **Plot Accuracy Comparison:** Create a bar plot to compare the accuracy of each model visually.



**6. Comparison Table and Visualization**

In this section, we display a summary table comparing all models based on accuracy, precision, recall, and F1-score.

**Steps:**

1. **Refine the Comparison Table:** Simulate results from the previous models and generate a table to compare all key metrics across models.

