Module 1 - Task 1

Load the train data.

- Import Pandas and alias it as 'pd'.
- Read the CSV file movies train.csv into a Pandas DataFrame named 'train'.
- To import the 'train.csv' file, which is located in the root path of your project, you should use the following path: './train.csv'.
- Inspect the data by calling the variable 'train'.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 1 - Task 2

Load the test data.

- Read the CSV file movies test.csv into a Pandas DataFrame named 'test'.
- To import the 'test.csv' file, which is located in the root path of your project, you should use the following path: './test.csv'.
- Inspect the data by calling the variable 'df'.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 1 - Task 3

Counting Null Values in train data.

- Calculate the count of null (missing) values in a DataFrame named 'train'.
- Use the isnull() method on the 'train' DataFrame to create a new DataFrame where each element is either True (if the corresponding element in 'train' is null) or False (if it's not null).
- Calculate the sum of True values for each column of the new DataFrame, effectively counting the number of null values in each column.
- Store the count of null values for each column in a Pandas Series named 'null_values_train'.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 1 - Task 4

Counting Null Values in test data.

- Calculate the count of null (missing) values in a Pandas DataFrame called 'test'.
- Use the .isnull() method on the 'test' DataFrame to create a new DataFrame of the same shape. Each element in this new DataFrame will be either True if the corresponding element in 'test' is null or False if it's not null.
- Calculate the sum of True values for each column of the new DataFrame. This effectively counts the number of null values in each column of the 'test' DataFrame.
- Store the count of null values for each column in a Pandas Series named 'null_values_test'.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 1 - Task 5

Handling Missing Values in train Data.

- Replace missing values (NA) in the 'Bed Grade' and 'City_Code_Patient' columns of the 'train' DataFrame using .fillna().
- Impute missing values with the mode (most frequent value) of the 'Bed Grade' and 'City_Code_Patient' columns.
- Apply the replacement directly to the 'train' DataFrame with inplace=True for both steps.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 1 - Task 6

Handling Missing Values in test Data.

- Replace missing values (NA) in the 'Bed Grade' and 'City_Code_Patient' columns of the 'test' DataFrame using .fillna().
- Impute missing values with the mode (most frequent value) of the 'Bed Grade' and 'City_Code_Patient' columns.
- Apply the replacement directly to the 'test' DataFrame with inplace=True for both steps.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 2 - Task 1

Encoding 'Stay' Column in train Data.

- Import the LabelEncoder class from the sklearn.preprocessing module.
- Create an instance of the LabelEncoder class and assign it to the variable 'le'.
- Use the fit_transform method of the LabelEncoder to encode the values in the 'Stay' column of the 'train' DataFrame. The .astype('str') ensures that the values are treated as strings.
- Update the 'Stay' column in the 'train' DataFrame with the transformed values. The original categorical values have now been replaced with numerical labels.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 2 - Task 2

Setting a Default Value.

- In the 'test' DataFrame, assign the value -1 to the 'Stay' column for all rows.
- This operation sets the 'Stay' column for all records in the 'test' DataFrame to have a constant value of -1.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 2 - Task 3

Merging Train and Test Data.

- Create a new DataFrame named 'df' using the `pd.concat` function.
- Concatenate the 'train' and 'test' DataFrames along their rows (vertically).
- Use the `ignore_index=True` parameter to reset the index of the combined DataFrame, ensuring a continuous, new index.
- The resulting 'df' DataFrame now contains all the data from 'train' and 'test' DataFrames in a single, merged DataFrame with a unified index.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 2 - Task 4

Categorical Data Label Encoding.

- Iterate through a list of categorical column names in the 'df' DataFrame: 'Hospital_type_code', 'Hospital_region_code', 'Department', 'Ward_Type', 'Ward_Facility_Code', 'Type of Admission', 'Severity of Illness', and 'Age'.
- For each column in the list, create a new instance of the LabelEncoder class and assign it to the variable 'le'.
- Use the fit_transform method of the LabelEncoder to encode the values in the current categorical column from the 'df' DataFrame. The .astype(str) ensures that the values are treated as strings.
- Update the column in the 'df' DataFrame with the transformed numerical labels, effectively replacing the original categorical values with encoded numerical values.
- This loop applies label encoding to convert categorical columns into numerical labels for all columns in the provided list.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 3 - Task 1

Filtering the Training Dataset.

- Create a new DataFrame called 'train' by filtering the existing DataFrame 'df'.
- The filter is based on a condition:
- This condition selects rows in the 'df' DataFrame where the value in the 'Stay' column is not equal to -1.
- The resulting 'train' DataFrame now contains only the rows where 'Stay' is not equal to -1.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 3 - Task 2

Filtering the Testing Dataset.

- Create a new DataFrame called 'test' by filtering the existing DataFrame 'df'.
- The filter is based on a condition:
- This condition selects rows in the 'df' DataFrame where the value in the 'Stay' column is equal to -1.
- The resulting 'test' DataFrame now contains only the rows where 'Stay' is equal to -1.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 3 - Task 3

Column Removal in the Test DataFrame.

- Create a new DataFrame called 'test1' as a subset of the 'test' DataFrame.
- Use the `drop` method to remove specific columns from 'test' based on the specified labels.
- The columns being dropped are: 'Stay', 'patientid', 'Hospital_region_code', and 'Ward_Facility_Code'.
- The `axis=1` parameter indicates that columns (not rows) should be dropped.
- The resulting 'test1' DataFrame is a modified version of 'test' with the specified columns removed.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 3 - Task 4

Column Removal in the Train DataFrame.

- Create a new DataFrame named 'train1' as a modified version of the 'train' DataFrame.
- Utilize the drop method to eliminate specific columns from the 'train' DataFrame based on the provided column labels.
- The columns being dropped are: 'case_id', 'patientid', 'Hospital_region_code', and 'Ward_Facility_Code'.
- The axis=1 parameter specifies that columns (not rows) should be removed from the DataFrame.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 4 - Task 1

Splitting Data into Training and Testing.

- Import the necessary module, train_test_split, from scikit-learn's model_selection library for splitting data into training and testing sets.
- Create feature and target variables:
- X1 represents the feature variables by excluding the 'Stay' column. The 'drop' method is used with axis=1 to remove the 'Stay' column.
- y1 represents the target variable and is set to the 'Stay' column.
- Split the data into training and testing sets using train_test_split:
- The feature variables are denoted as X1, and the target variable is denoted as y1.
- Use the test_size parameter to specify the proportion of data to be allocated for testing. In this case, it's set to 20% of the data.
- The random_state parameter(random_state = 100) sets a seed for randomization, ensuring reproducibility of the split.
- After the split, the variables X_train, X_test, y_train, and y_test hold the feature and target data for both the training and testing sets.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 4 - Task 2

Training an XGBoost Classifier and Evaluating Accuracy.

- Import the necessary libraries and modules, including xgboost for XGBoost, and accuracy_score from scikit-learn for evaluating the model's accuracy.
- Create an XGBoost classifier instance named 'classifier_xgb' with specified hyperparameters:
- max_depth=4: Maximum depth of the tree.
- learning_rate=0.1: Learning rate for gradient boosting.
- n_estimators=800: The number of boosting rounds (trees).
- objective='multi:softmax': Objective function for multi-class classification.
- reg_alpha=0.5 and reg_lambda=1.5: L1 and L2 regularization terms.
- booster='gbtree': Type of boosting model.
- n_jobs=4: Number of CPU cores used for parallel processing.
- min_child_weight=2: Minimum sum of instance weight (hessian) needed in a child.
- base_score=0.75: Initial prediction made by the model.
- Fit the 'classifier_xgb' model to the training data ('X_train' and 'y_train') and assign it to 'model_xgb'.
- Use the trained 'model_xgb' to make predictions on the test data ('X_test') and store the predictions in 'prediction_xgb'.
- Calculate the accuracy of the XGBoost model's predictions by comparing 'prediction_xgb' with the true test labels ('y_test') using the accuracy_score function. The accuracy score is stored in 'acc_score_xgb'.
- Round the accuracy score to two decimal places.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 4 - Task 3

Post-processing Model Predictions.

- Use the trained 'classifier_xgb' model to make predictions on the 'test1' DataFrame. The 'predict' method is used to obtain predictions, and these predictions are stored in 'pred_xgb'.
- Exclude the first column(case_id) from the 'test1' when making predictions.
- Create a new DataFrame named 'result_xgb' to organize the prediction results. The 'pred_xgb' values are added to this DataFrame with the column name 'Stay'.
- Assign the 'case_id' column from 'test1' to the 'case_id' column in the 'result_xgb' DataFrame.
- Reorder the columns in 'result_xgb' to have 'case_id' as the first column and 'Stay' as the second column.
- Replace the numeric labels in the 'Stay' column of 'result_xgb' with meaningful categories using the .replace() method. The numeric labels are mapped to their corresponding 'Stay' categories, making the results more interpretable.
- The 'result_xgb' DataFrame now contains the 'case_id' and 'Stay' columns with predictions converted into human-readable stay categories.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.

Module 4 - Task 4

Counting Unique Cases per Stay Category.

- Utilize the 'result_xgb' DataFrame to perform a grouping operation based on the 'Stay' column.
- Group the data by unique 'Stay' values using the groupby method.
- Calculate the number of unique 'case_id' values in each 'Stay' group by applying the .nunique() method to the 'case_id' column.
- The result is stored in the variable 'result', which is a Pandas Series containing the count of unique 'case_id' values for each 'Stay' category.

Helpful Links:

To complete this task, knowledge of Machine Learning is required. You can learn Python and ML using the following resources:

- Machine Learning Course
- Python Interview QnA
- Python Coding Practice

Note: Complete the code, and confirm success with "Run Test".
Jupyter Notebook Trick: If you wish to collapse and make the output area scrollable for the code cell associated with this task, simply click on the white area located to the left of the output.