Module 1 - Task 1

Load the data.

- Import Pandas and alias it as 'pd'.
- Read the CSV file movies insurance.csv into a Pandas DataFrame named 'df'.
- To import the 'insurance.csv' file, which is located in the root path of your project, you should use the following path: './insurance.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 2

Finding Duplicates.

- Calculate the number of duplicate rows in the DataFrame 'df' using the duplicated() method and then sum them up using the sum() method.
- Display the total number of duplicate rows, which is stored in the variable 'duplicates'.

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

Removing Duplicate Rows.

- Apply the drop_duplicates method to 'df' to remove duplicate rows.
- The inplace=True argument is used to modify 'df' in place, which means it will remove duplicates directly from 'df' without the need to assign the result to a new variable.
- After executing this code, 'df' will be updated with the duplicate rows 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 1 - Task 4

Counting Null Values.

- Apply the .isnull() method to 'df' to identify and mark null values, returning a DataFrame with True/False values.
- Use the .sum() method on the result to count the number of null values in each column.
- Store the count of null values in the variable 'null_values'.

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

Label Encoding of Categorical Features.

- Import the LabelEncoder class from the sklearn.preprocessing module.
- Create an instance of the LabelEncoder class and assign it to the variable lab_encode.
- The LabelEncoder is used to convert categorical variables into numeric form for machine learning algorithms.
- Loop through the columns "sex" and "smoker" in the DataFrame df and apply label encoding to them. The loop iterates through the column names, not the actual data.
- For each column (e), call the fit_transform method of the lab_encode object to encode the values in that column.

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

One-Hot Encoding of Categorical Data.

- Use the 'pd.get_dummies()' function to perform one-hot encoding specifically on the "region" column of the DataFrame 'df'.
- The result is stored in the 'one_hot_encode' DataFrame.

Hint: To perform one-hot encoding for a specific column, use `df['column_name']` inside the method and avoid using any additional parameters.

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

Concatenating a DataFrame with One-Hot Encoded Columns.

- Use the 'pd.concat()' function to concatenate the original DataFrame 'df' with another DataFrame 'one_hot_encode'.
- The 'pd.concat()' function is used for combining two or more DataFrames along a particular axis. In this case, 'axis=1' means the DataFrames will be concatenated horizontally (adding columns side by side).
- The result of this concatenation is stored in the 'df1' 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 2 - Task 4

Dropping a Column.

- Use the 'drop' method on the DataFrame 'df1' to remove a specific column, "region", from the DataFrame.
- The 'axis=1' argument is provided to indicate that the operation should be performed along the columns, meaning a column will be dropped.
- The 'inplace=True' argument is set to modify the DataFrame 'df1' directly, so there's no need to assign the result to a new variable. This makes the change permanent.

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

Splitting a DataFrame into Train and Test Sets for Machine Learning.

- Import the train_test_split function from the sklearn.model_selection module. This function is commonly used to split a dataset into training and testing subsets.
- Create a DataFrame 'X' by dropping the "charges" column from the DataFrame 'df1'. The 'X' DataFrame will contain the feature variables used for prediction.
- Create a Series 'y' by selecting the "charges" column from the DataFrame 'df1'. This 'y' variable represents the target variable to be predicted.
- Use the 'train_test_split' function to split the dataset into training and testing sets. 'X' represents the feature data, 'y' represents the target data, and 'test_size=0.2' specifies an 80-20 split, with 80% of the data used for training and 20% for testing.
- The 'random_state' argument is set to '42' to ensure reproducibility; the same split will occur each time you run the code with this value.
- After splitting, you have four variables: 'X_train' (the training features), 'X_test' (the testing features), 'y_train' (the training target values), and 'y_test' (the testing target values).

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

Evaluating Random Forest Regression Model.

- Import the numpy library as 'np' to perform numerical operations and the necessary libraries for machine learning tasks, including 'cross_val_score' and 'RandomForestRegressor'.
- Create an instance of the 'RandomForestRegressor' class and assign it to 'rand_forest_model'. This model is a random forest regressor with specific configurations.
- 'n_estimators=50' indicates that the random forest will consist of 50 decision trees.
- 'n_jobs=2' specifies that two CPU cores should be used for parallel processing.
- 'random_state=42' sets the random seed for reproducibility.
- Use 'cross_val_score' to evaluate the performance of the 'rand_forest_model'. It takes the following arguments:
- 'rand_forest_model': The machine learning model to be evaluated.
- 'X_train': The training feature data.
- 'y_train': The corresponding target values.
- 'scoring="neg_mean_squared_error"': The scoring metric used to assess model performance, which is the negative mean squared error.
- 'cv=10': The number of cross-validation folds, which is set to 10. Cross-validation helps assess a model's generalization performance.
- The 'cross_val_score' function returns an array of negative mean squared error scores for each fold of the cross-validation.
- Calculate 'performance' by taking the square root of the negative mean squared error scores to obtain root mean squared error (RMSE) values, which are more interpretable.
- Compute the standard deviation ('std') of the RMSE values, which provides information about the variability of model performance across different cross-validation folds. A lower standard deviation indicates more consistent model performance.
- 'std' represents how much the model's performance varies across the different folds of the cross-validation, giving insight into the model's stability and robustness.

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

Training a Model, Making Predictions, and Comparing Results.

- The model is being trained using the training dataset, 'X_train' for features and 'y_train' for target values.
- After training, the model is ready to make predictions on new data.
- 'predictions' is created by using the trained model to predict target values for the 'X_test' dataset, which is a set of feature data from the testing dataset.
- To make the predicted values more readable, 'rounded_predictions' is created by rounding the 'predictions' to two decimal places using the 'np.round()' function.
- A comparison of the first 10 entries of the actual charges ('y_test' - it should be also rounded to 2 decimal places) and the predicted charges ('rounded_predictions') is created in a DataFrame called 'compare'.
- The 'compare' DataFrame displays both the actual and predicted charges for the first 10 instances, allowing for a quick visual assessment of how well the model's predictions align with the actual values.
- This type of comparison is valuable for understanding the model's accuracy on unseen data.

Note: Make sure to use .values when slicing the y_test data to ensure we're working with a NumPy array instead of a pandas Series.

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.