1. Encoding the dataset using map:

```
dataset.loc[:, -1] = dataset.loc[:, -1].map(
  {species: index for index, species in enumerate(dataset.loc[:, -1].unique())}
)
```

Atikul Islam Sajib

2. Create Custom Class Exception:

```
class ModelNotDefinedError(Exception):
    """Exception raised when the model is not defined."""

    def __init__(self, message="Model is not defined."):
        self.message = message
        super().__init__(self.message)
```

Atikul Islam Sajib

3. Lambda function used to filter the target class:

```
    def filter_target_class(self, target_class):
        # Return the filter by target class
        return lambda data: data[1] in target_class
```

Atikul Islam Sajib

4. Define Epochs using list comprehension:

```
    def epoch_initialization(self, num_values):
        start = 0
        end = 10
        return [int(start + (end - start) *i /(num_values -1)) for i in range(num_values)]
```

Atikul Islam Sajib

5. save each epoch result in a file:

```
    with open('epoch_based_result.txt', 'a') as f:
        """Save the epoch based result into the text file"""
        f.write(str(trainer.train()))

    """Close the file"""
    f.close()
```

Atikul Islam Sajib

6. Scaling the dataset with proper function name:

```
    def data_scaling(df):
        scaler = StandardScaler()
        transform_df = scaler.fit_transform(df.iloc[:, :-1])
        independent = pd.DataFrame(transform_df)
        dependent = pd.DataFrame(df.iloc[:, -1])
        new_df = pd.concat([independent, dependent], axis=1)

        return new_df
```

Atikul Islam Sajib

7. total trainable parameters using sum()

```
    total_trainable_parameters = sum(total_params))
```

Atikul Islam Sajib

8. Import metrics with proper format align:

```
        from sklearn.metrics import (
            accuracy_score,
            precision_score,
            recall_score,
            f1_score,
            classification_report
    )
```

Atikul Islam Sajib

9. Unit Testing:

```
def test_evaluation_with_invalid(self):
    """
    Test the evaluation of the trained model with invalid data
    """
    self.test_loader = not None
    with self.assertRaises(Exception):
        self.trainer.model_evaluate(dataloader=self.test_loader, model=self.model)
```

Atikul Islam Sajib

10.Radian to degree:

```
math.degrees(math.acos(rad))
```

Atikul Islam Sajib