# CSE 676-B: Deep Learning, Spring 2024

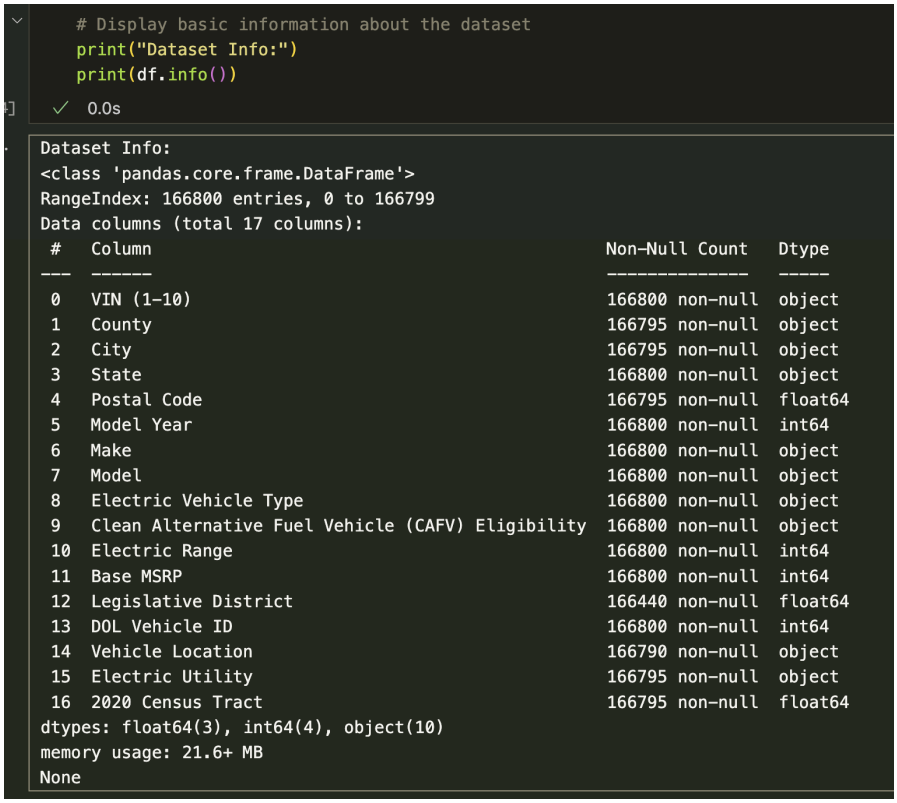
Assignment 0 From Data to ML and NN Models

## 1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? Provide the main statistics about the entries of the dataset (mean, std, number of missing values, etc.)

This dataset contains information about the Battery Electric Vehicles (BEVs), Plug-in Hybrid Electric Vehicles (PHEVs), Make, Model, Year, Range, and Clean Alternative Fuel Vehicle (CAFV) Eligibility that are currently registered through the Washington State Department of Licensing (DOL).

The dataset comprises a mix of categorical and numerical data types.

## Info of dataset:



```
    # Display basic information about the dataset
    print("Dataset Info:")
    print(df.info())
```

✓ 0.0s

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 166800 entries, 0 to 166799
Data columns (total 17 columns):
 #   Column                                             Non-Null Count   Dtype
---  ------                                             --------------   -----
 0   VIN (1-10)                                         166800 non-null  object
 1   County                                             166795 non-null  object
 2   City                                               166795 non-null  object
 3   State                                              166800 non-null  object
 4   Postal Code                                        166795 non-null  float64
 5   Model Year                                         166800 non-null  int64
 6   Make                                               166800 non-null  object
 7   Model                                              166800 non-null  object
 8   Electric Vehicle Type                              166800 non-null  object
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  166800 non-null  object
 10  Electric Range                                     166800 non-null  int64
 11  Base MSRP                                          166800 non-null  int64
 12  Legislative District                               166440 non-null  float64
 13  DOL Vehicle ID                                     166800 non-null  int64
 14  Vehicle Location                                   166790 non-null  object
 15  Electric Utility                                   166795 non-null  object
 16  2020 Census Tract                                  166795 non-null  float64
dtypes: float64(3), int64(4), object(10)
memory usage: 21.6+ MB
None
```

Summary statistics of the dataset:

```
# Display summary statistics of the dataset
print("\nSummary Statistics:")
print(df.describe())
```

[5] ✓ 0.0s

```
Summary Statistics:
         Postal Code     Model Year  Electric Range      Base MSRP  \
count  166795.000000  166800.000000   166800.000000  166800.000000
mean    98173.713750    2020.341793       61.508993    1152.723171
std      2442.584415       3.001465       93.271747    8661.081091
min      1730.000000    1997.000000        0.000000       0.000000
25%     98052.000000    2018.000000        0.000000       0.000000
50%     98122.000000    2021.000000        0.000000       0.000000
75%     98371.000000    2023.000000       84.000000       0.000000
max     99577.000000    2024.000000      337.000000  845000.000000

       Legislative District  DOL Vehicle ID  2020 Census Tract
count         166440.000000    1.668000e+05       1.667950e+05
mean              29.178941    2.172420e+08       5.297709e+10
std               14.853534    7.727458e+07       1.569754e+09
min                1.000000    4.385000e+03       1.001020e+09
25%               18.000000    1.790741e+08       5.303301e+10
50%               33.000000    2.244045e+08       5.303303e+10
75%               42.000000    2.513421e+08       5.305307e+10
max               49.000000    4.792548e+08       5.603300e+10
```

Number of missing values:

Total rows - rows after deleting NA values = 166800 - 166435 = **365**

## 2. What kind of preprocessing techniques have you applied to this dataset?

Dropped missing values rows and duplicates.

```
# Drop rows with missing values
df.dropna(inplace=True)

# Remove duplicates
df.drop_duplicates(inplace=True)
```

✓ 0.1s

Dropped the features which are not required or not relevant to the dataset. Also renamed large feature names to small readable ones.

```
columns_to_drop = ['VIN (1-10)', 'State', 'Postal Code', 'Base MSRP', 'Legislative District',
                   'DOL Vehicle ID', 'Vehicle Location', 'Electric Utility', '2020 Census Tract']
df.drop(columns=columns_to_drop, inplace=True, axis=1)

new_names = {'Model Year': 'Model_Year', 'Electric Vehicle Type': 'EV_Type',
             'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 'CAFV_Eligibility', 'Electric Range': 'Rang
df.rename(columns=new_names, inplace=True)
```

✓ 0.0s Python

There were only two numerical features named Range and Make Year, which contained real-world values with no outliers.

Performed One-Hot encoding using Pandas Dummies.

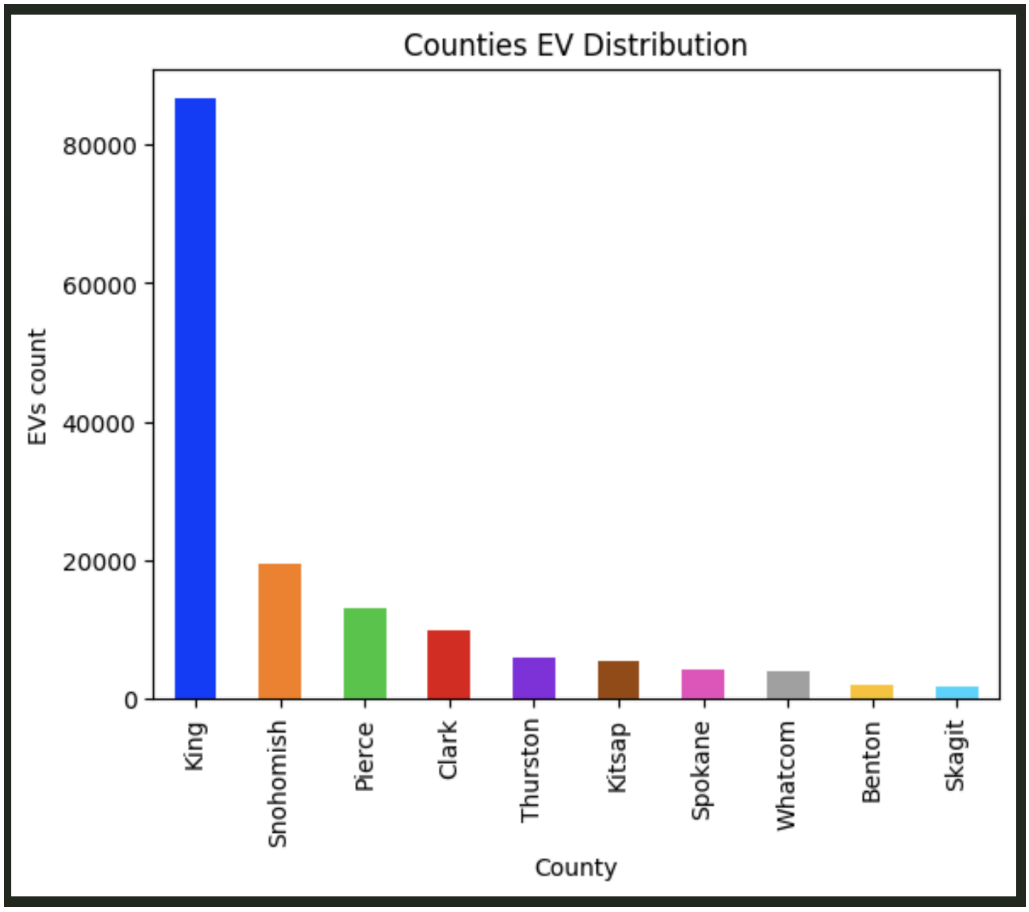
```
# One-Hot Encoding
df_encoded = pd.get_dummies(
    df, columns=['CAFV_Eligibility', 'EV_Type'], prefix=['CAFV', 'EV'])

columns_to_convert = ['CAFV_Clean Alternative Fuel Vehicle Eligible', 'CAFV_Eligibility unknown as battery ra
                      'CAFV_Not eligible due to low battery range', 'EV_Battery Electric Vehicle (BEV)', 'EV_

# Convert boolean values to integers (0 and 1)
df_encoded[columns_to_convert] = df_encoded[columns_to_convert].astype(int)
```

✓ 0.0s Python

+ Code + Markdown

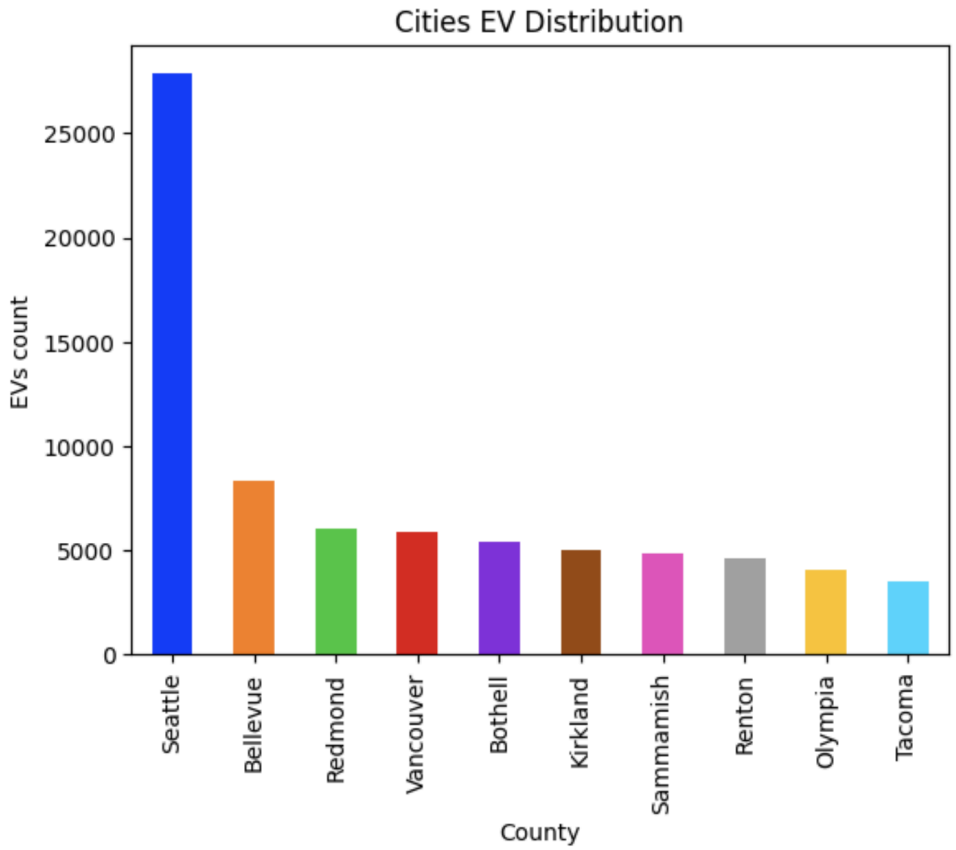
3. Provide at least 5 visualization graphs with a brief description for each graph, e.g. discuss if there are any interesting patterns or correlations.

Top ten counties with the highest number of EVs.



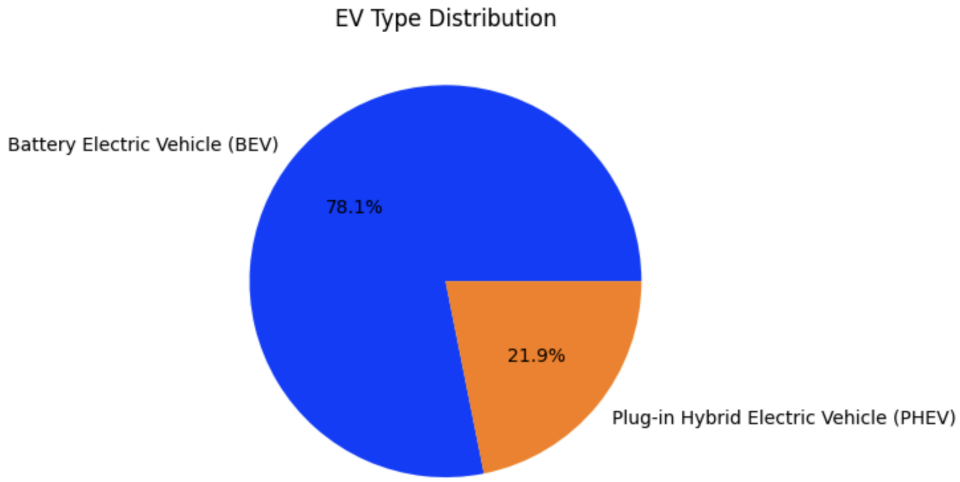

→ King County has the highest number of Electric vehicles in Washington State.

## Top ten Cities with the highest number of EVs.



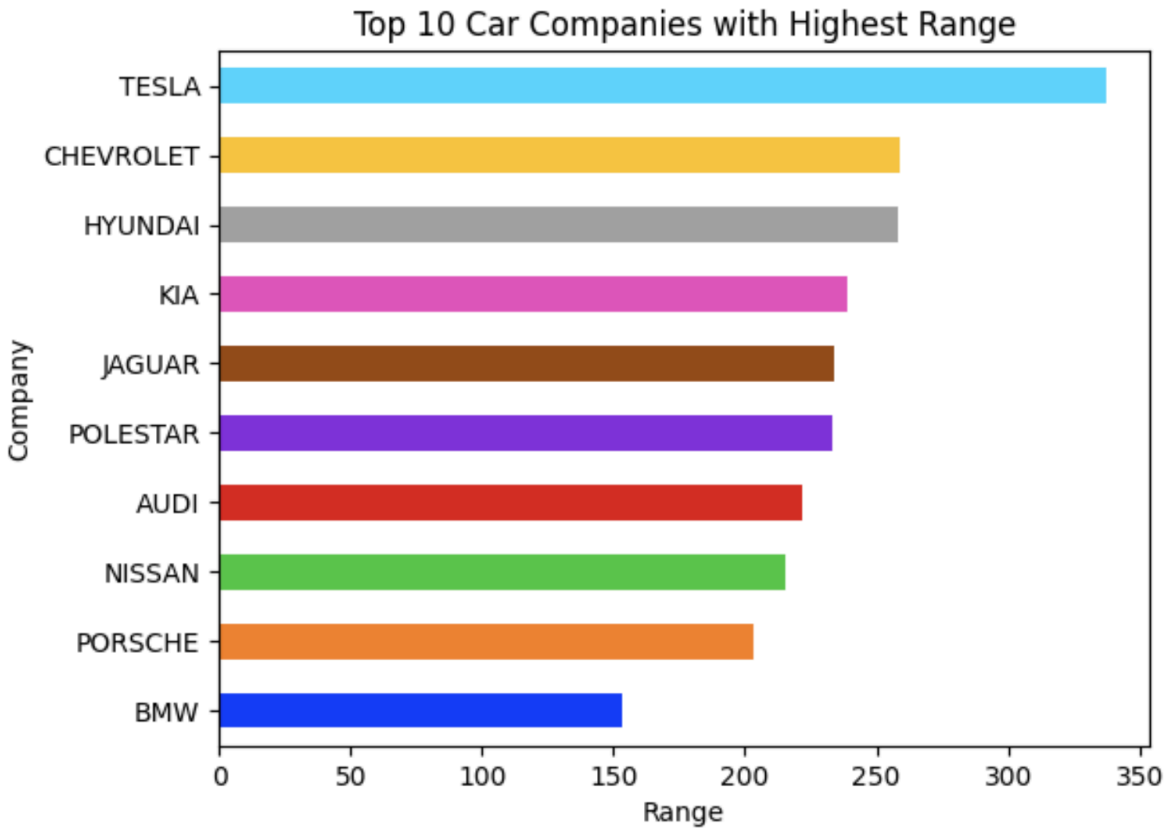

→ Seattle City has the highest number of Electric vehicles in Washington State.

## EV Type Distribution.



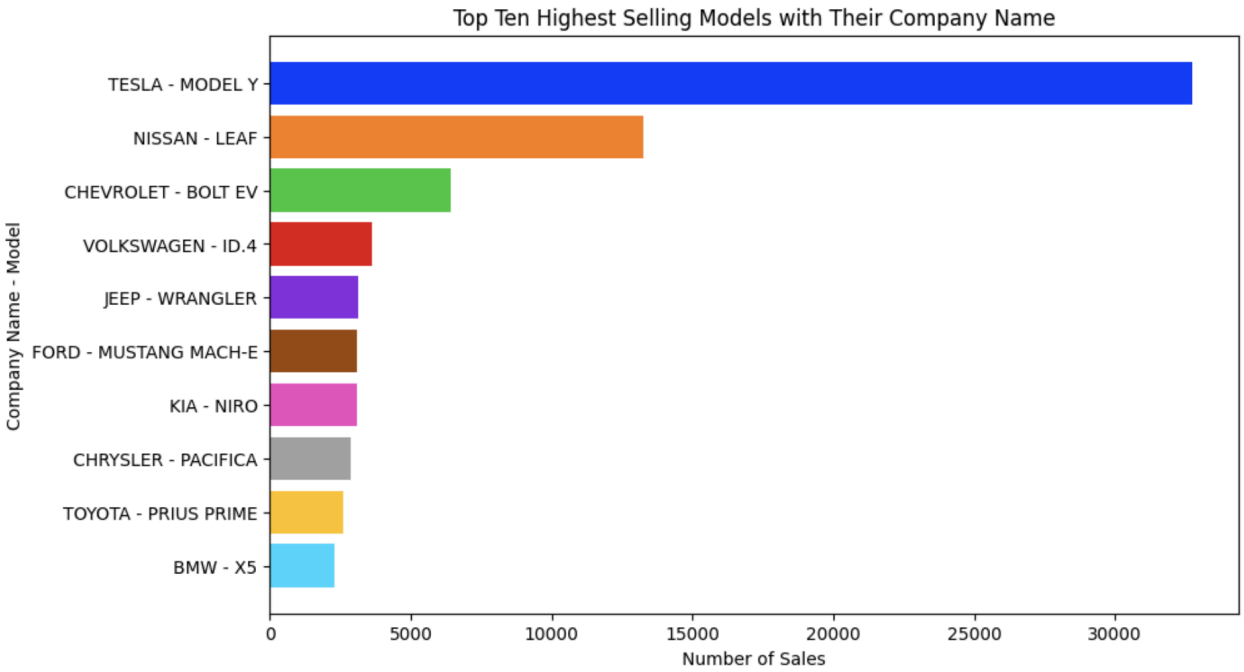
→ Battery-operated EVs are much higher in number compared to Plug-In EVs.

## Top ten car companies with the highest range.



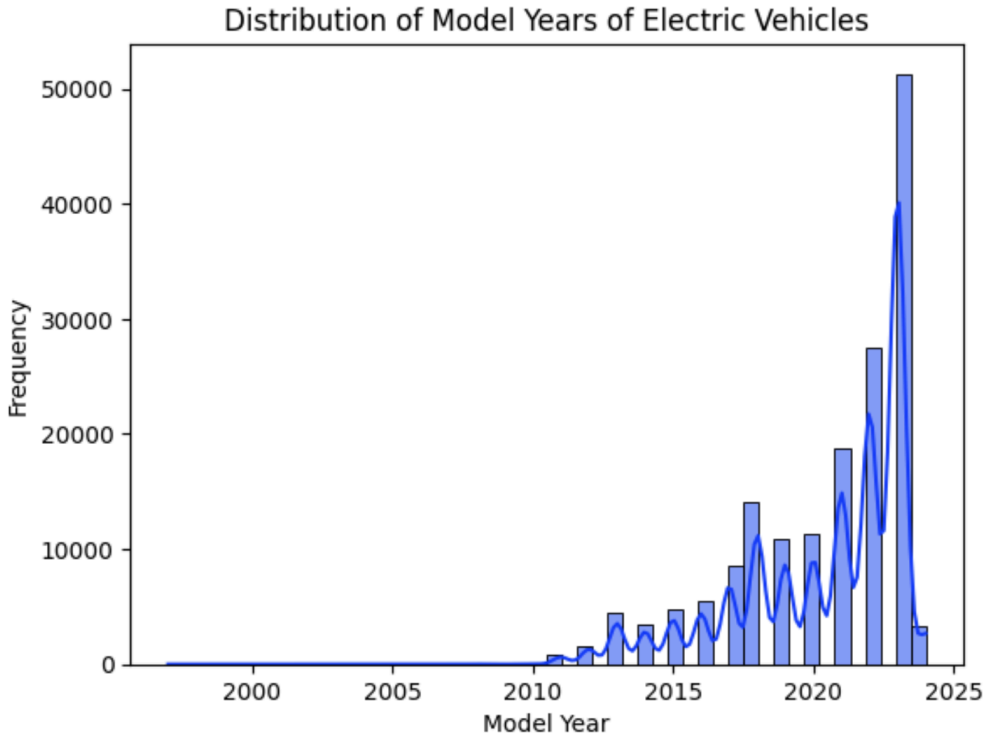

→ Tesla has the highest range of all-electric vehicles.

Top ten highest-selling models with their company name.



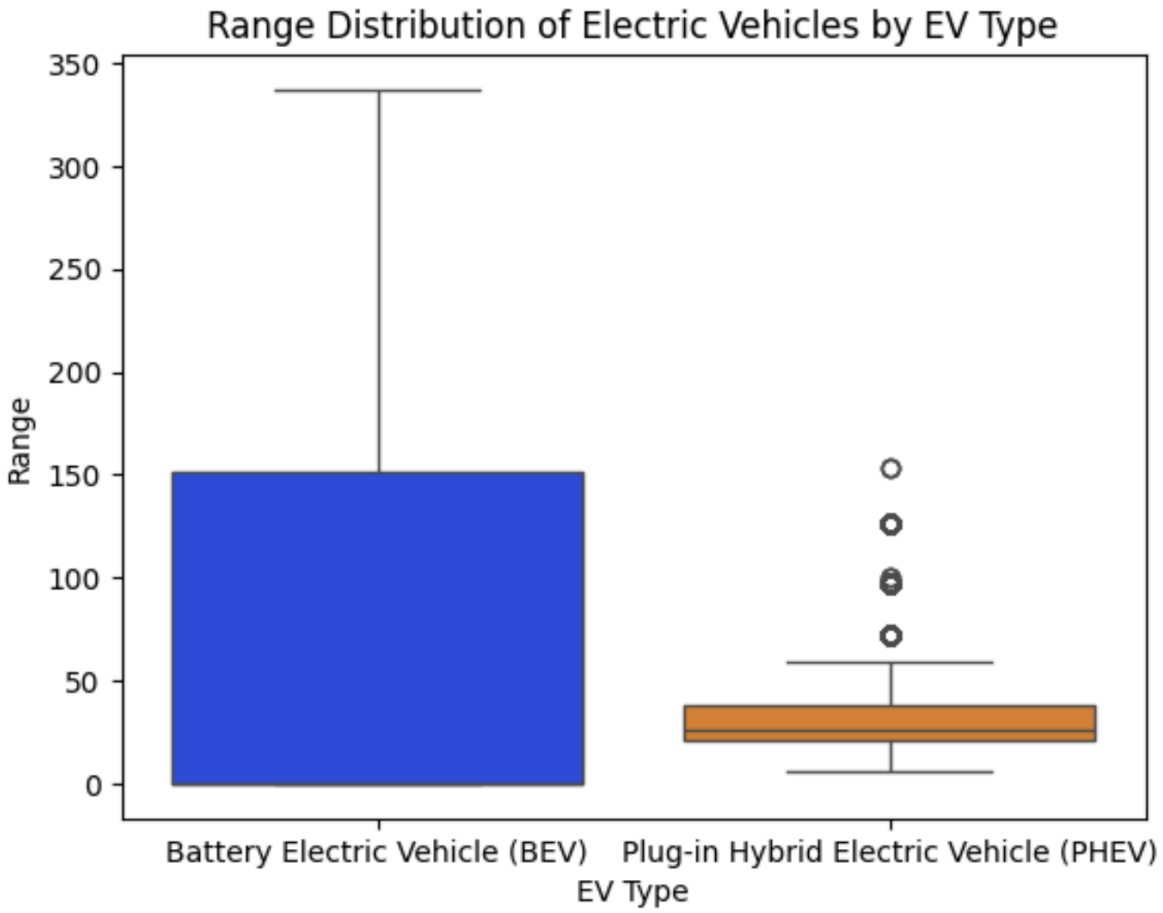

→ Tesla's Model-Y is the most selling EV of all time.

## Distribution of Model Years of EVs.



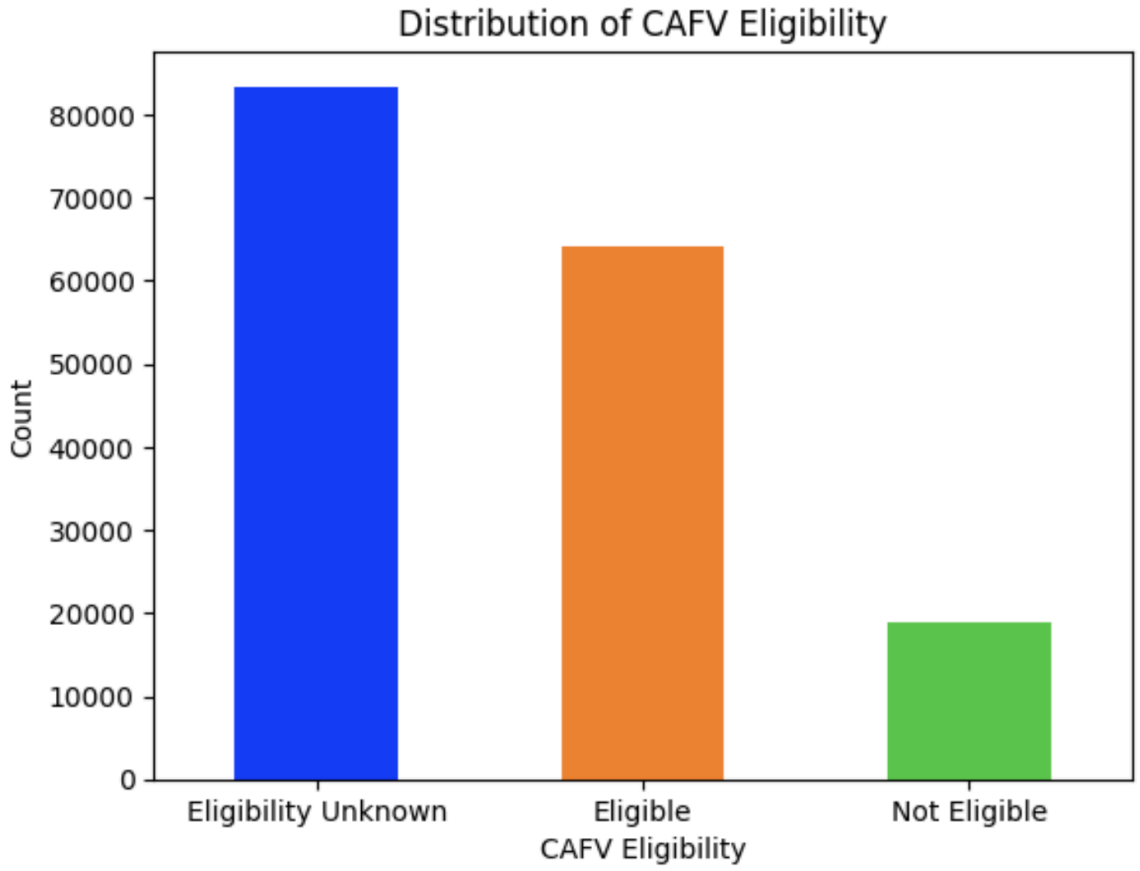

→ EV sales saw a steep growth from the year 2020-2025.

Range distribution of Electric Vehicles by EV types.



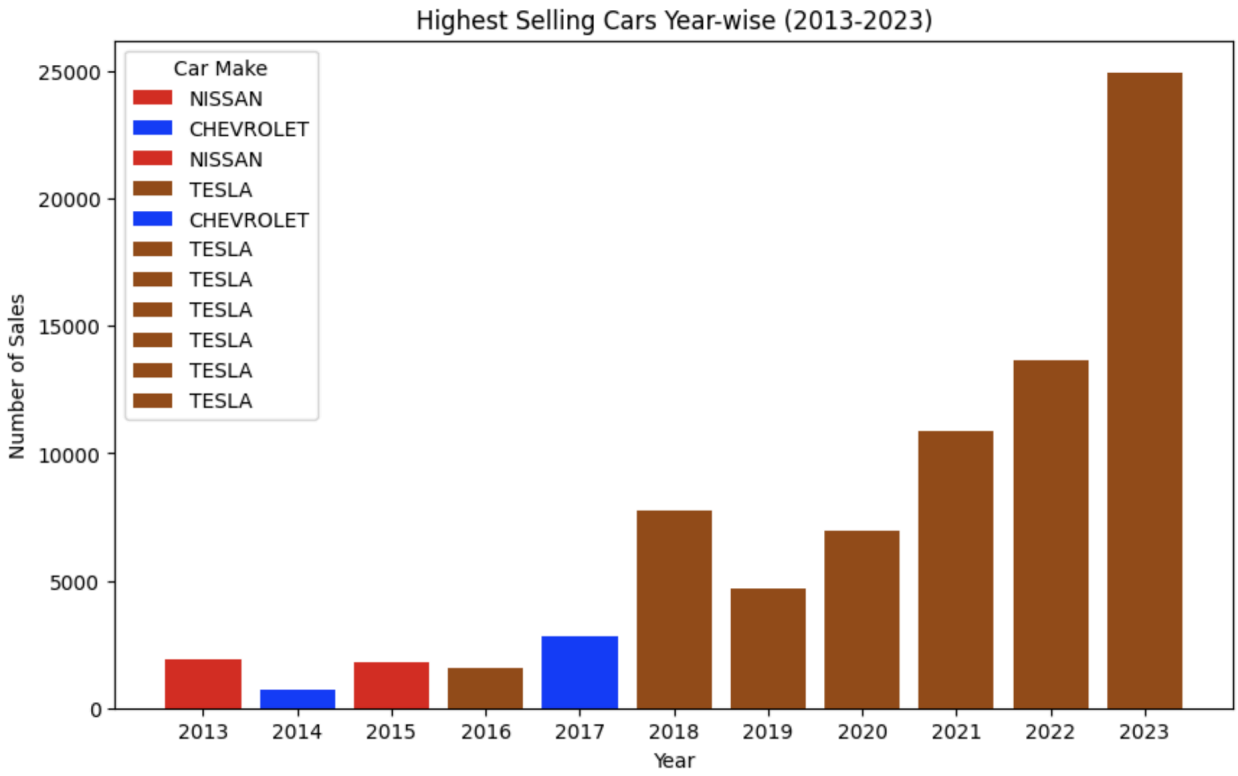

→ From above we can infer that BEV has a much higher range than PHEV.

Distribution of CAFV Eligibility.



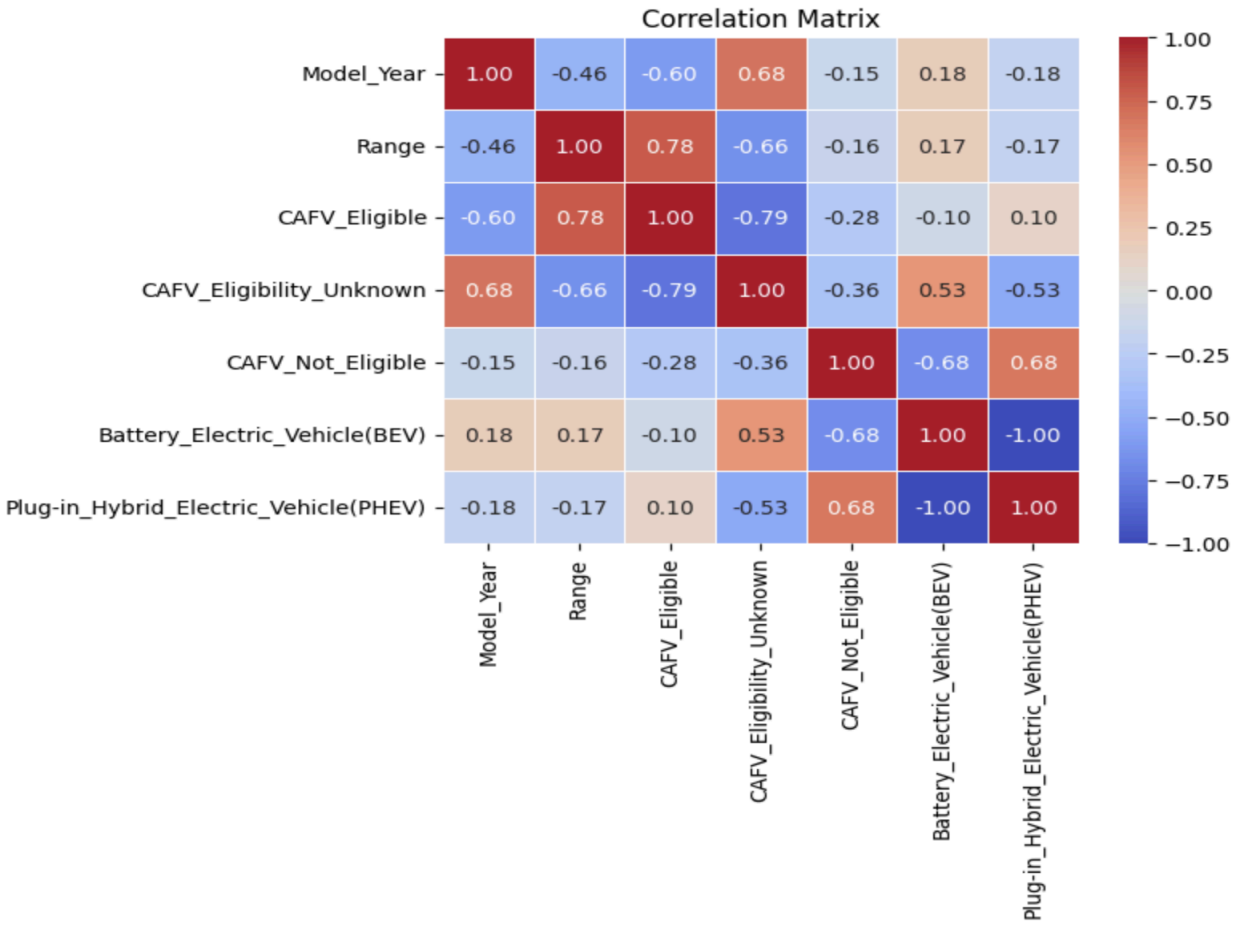

→ Most of the EVs in Washington state have their CAFV eligibility unknown.

## Highest selling car year-wise in last decade.




→ From the year 2018-2023, Tesla has been the highest-selling car in Washington state.

## Correlation Matrix.




→ Range and CAFV_Eligible have the highest correlation among all the features.

# 4. Provide brief details and mathematical representation of the ML methods you have used. What are the key features? What are the advantages/disadvantages?

## Logistic Regression:

Logistic Regression is a binary classification algorithm used to predict the probability of a binary result based on one or more features. In short classification. It uses the logistic function to find the probability of the dependent variable as a function of the independent variables.

**Key Features:**

It’s a simple, efficient classification model and also works well with linearly separable data.

Its advantage is it’s easy to implement and interpret however is sensitive to outliers.

## Random Forest:

Random Forest is an ensemble learning method that constructs multiple decision trees during training. It builds multiple decision trees and merges them to get accurate and stable predictions.

**Key Features:**
It handles non-linear relationships better and since it’s an ensemble learning, it combines multiple weak learners to create a strong learner.

Its advantage is it handles high dimensional data well however, it’s computationally expensive.

## Support Vector Machines(SVM):

It’s an algorithm used for both classification and regression tasks. It works by finding the hyperplane that best separates the classes in the feature space. SVM aims to find the optimal hyperplane that maximizes the margin between the classes in the feature space.

**Key Features:**
It handles both linear and non-linear relationships better and is effective in high-dimensional data.

Its advantage is it handles high-dimensional data well however, it’s sensitive to noise.

5. Provide your loss value and accuracy for all 3 methods.

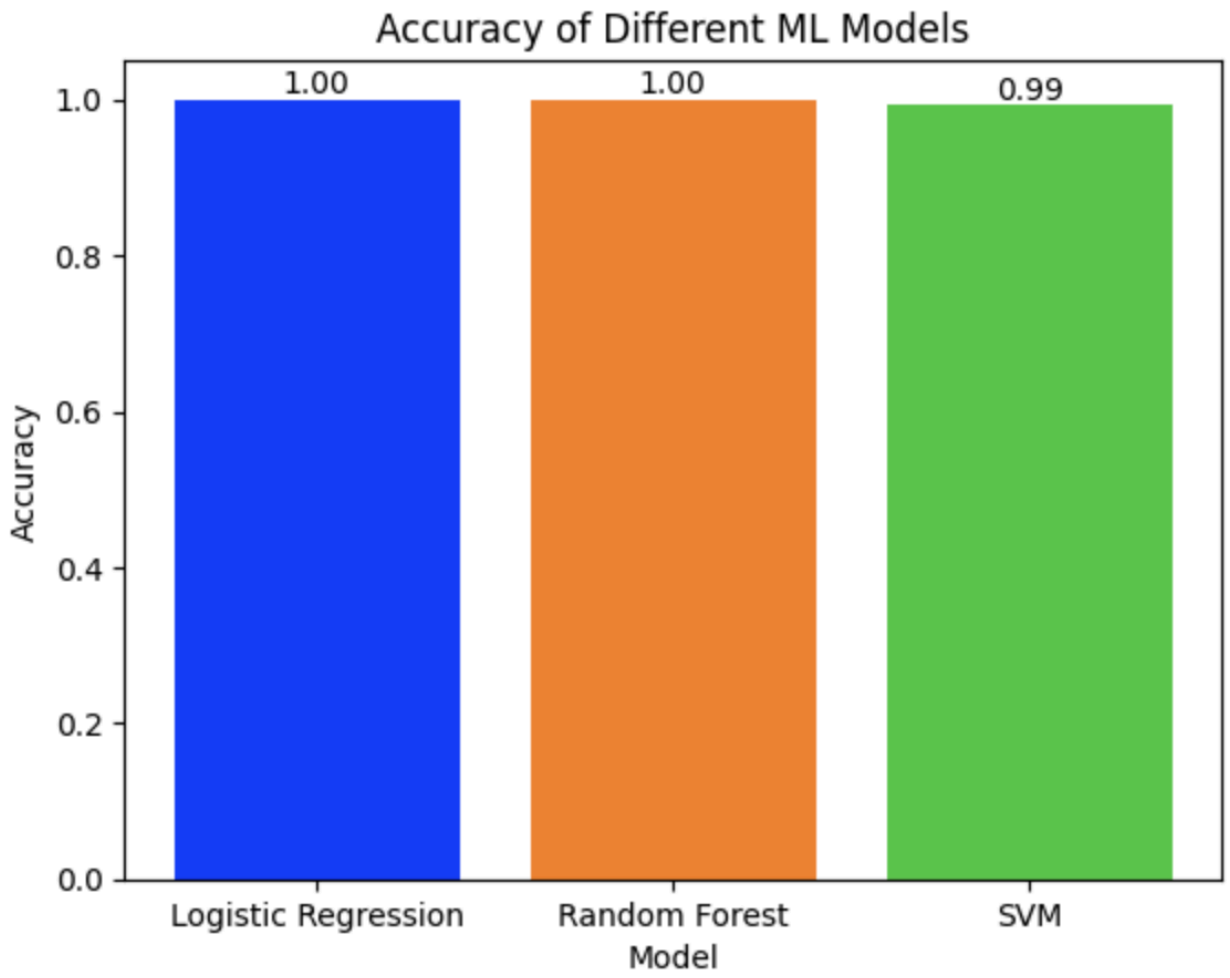
Validation Set

```
--- Validation Set Loss ---
Logistic Regression Loss on Validation Set: 2.2204460492503136e-16
Support Vector Machine(SVM) Loss on Validation Set: 0.6242489786109109
--- Random Forest Classification Report on Validation Set---
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10231
           1       1.00      1.00      1.00      6413

    accuracy                           1.00     16644
   macro avg       1.00      1.00      1.00     16644
weighted avg       1.00      1.00      1.00     16644

--- Validation Set ---
Logistic Regression Accuracy on Validation Set: 1.0
Random Forest Accuracy on Validation Set: 1.0
Support Vector Machine(SVM) Accuracy on Validation Set: 0.9904470079307859
```

Testing Set

```
--- Testing Set Loss ---
Logistic Regression Loss on Testing Set: 2.2204460492503136e-16
Support Vector Machine(SVM) Loss on Testing Set: 0.6269903262632939
--- Random Forest Classification Report on Testing Set---
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10307
           1       1.00      1.00      1.00      6336

    accuracy                           1.00     16643
   macro avg       1.00      1.00      1.00     16643
weighted avg       1.00      1.00      1.00     16643


--- Testing Set ---
Logistic Regression Accuracy on Testing Set: 1.0
Random Forest Accuracy on Testing Set: 1.0
Support Vector Machine(SVM) Accuracy on Testing Set: 0.9923090788920267
```

6. Show the plot comparing the predictions vs the actual test data for all methods used. Analyze the results. You can consider accuracy/time/loss as some of the metrics to compare the methods.



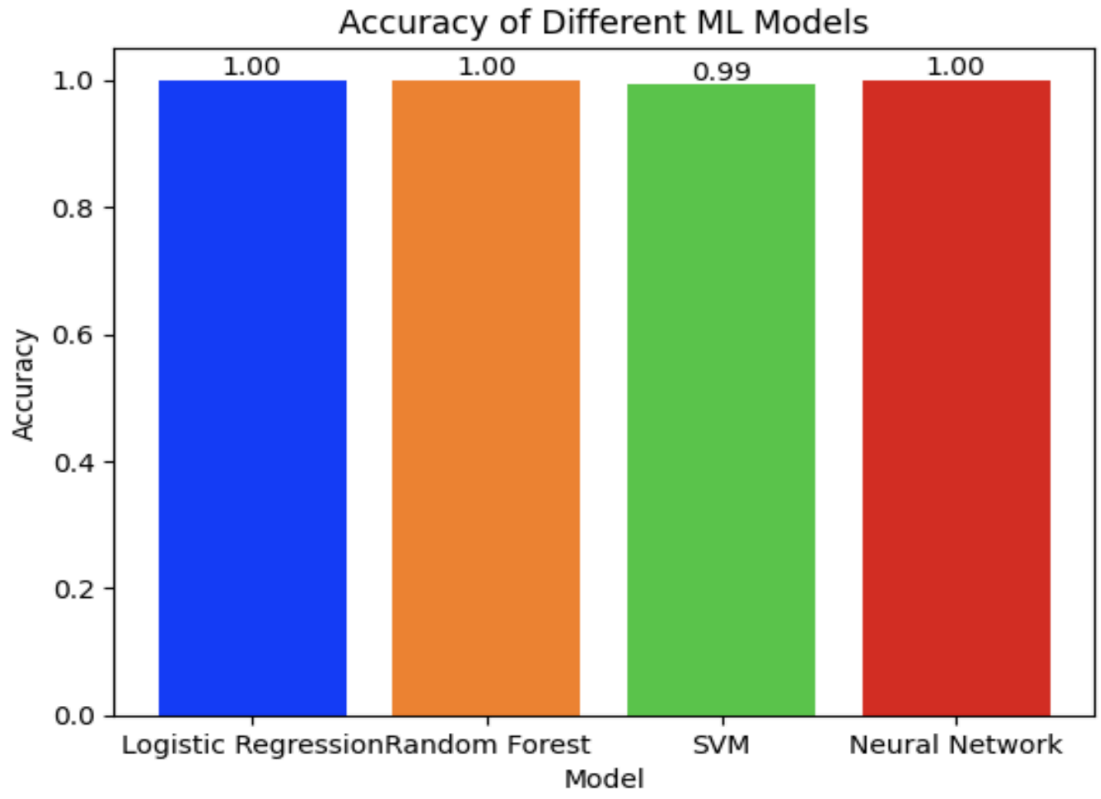

→ All three models Logistic Regression, Random Forest, and SVM gave similar accuracies.

7. Provide the neural network structure you have built to solve the problem defined in Part I. Show the plot. Analyze the results.

Neural Network Structure.

```python
# Define the neural network architecture
class NN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(NN, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

Accuracies of different models.




→ All models Logistic Regression, Random Forest, SVM, and Neural Network gave similar accuracies.

# Part III: OCTMNIST Classification

## 1. Describe the NN you have defined.

```
# Torchinfo Summary of base model
torchinfoSummary = torchinfo.summary(CNN(dropoutAdd=False), input_data=torch.randn(
    batch_size, 1, 28, 28))  # batch_size=32, channels=1, height=28, width=28
print(torchinfoSummary)
```

✓ 0.0s

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNN                                      [32, 4]                   --
├─Conv2d: 1-1                            [32, 16, 24, 24]          416
├─MaxPool2d: 1-2                         [32, 16, 12, 12]          --
├─Conv2d: 1-3                            [32, 32, 8, 8]            12,832
├─MaxPool2d: 1-4                         [32, 32, 4, 4]            --
├─Linear: 1-5                            [32, 512]                 262,656
├─Linear: 1-6                            [32, 256]                 131,328
├─Linear: 1-7                            [32, 4]                   1,028
==========================================================================================
Total params: 408,260
Trainable params: 408,260
Non-trainable params: 0
Total mult-adds (M): 46.59
==========================================================================================
Input size (MB): 0.10
Forward/backward pass size (MB): 3.08
Params size (MB): 1.63
Estimated Total Size (MB): 4.81
==========================================================================================
```

- **Input Layer:** This neural network takes input data in the form of 2D images with a single channel i.e. grayscale images.

- **Convolutional Layers:** The network starts with two convolutional layers (conv1 and conv2), which aim to identify patterns and features within the input images.

- **Pooling Layers:** After each convolutional layer, a pooling layer is applied. Therefore, we have two max pool layers. This layer reduces the dimensions of the feature maps outputted by the convolutional layers; thus helping to decrease the computational complexity and control overfitting.

- **Fully Connected Layers (fc1, fc2, and fc3):** Following the convolutional and pooling layers, there are three fully connected layers (fc1, fc2, and fc3). These layers act as

traditional neural network layers, where each neuron is connected to every neuron in the previous and subsequent layers. These layers help in learning complex patterns and relationships in the extracted features.

- **Activation Functions:** Between each layer, the Rectified Linear Unit (ReLU) activation function is applied which introduces non-linearity, allowing the network to learn complex mappings between the inputs and outputs.

- **Dropouts:** The network includes dropout layers after the first and second fully connected layers if dropoutAdd is set to True during initialization. Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of input units to zero during training.

- **Output Layer:** The final fully connected layer (fc3) produces the output of the network. The output layer has four neurons, each representing a class.

## 2. Describe how the techniques (regularization, dropout, early stopping) have impacted the performance of the model.

### Base Model

Accuracy on the testing dataset = 87.95
Precision = 0.87
Recall = 0.88
Fscore = 0.87

### Model with Regularization

Accuracy on the testing dataset = 85.55
Precision = 0.84
Recall = 0.86
Fscore = 0.84

### Model with Regularization and Dropout

Accuracy on the testing dataset = 86.78
Precision = 0.85
Recall = 0.87
Fscore = 0.85

## Model with Regularization, Dropout and Early Stopping

Accuracy on the testing dataset = 84.23
Precision = 0.82
Recall = 0.84
Fscore = 0.81

**L2 Regularization:**
L2 regularization penalizes large weights in the network by adding a term proportional to the squared magnitude of weights to the loss function. The second model, which includes L2 regularization, shows a decrease in performance compared to the base model. This is likely because L2 regularization might have overly penalized the model, causing it to generalize less effectively.

**Dropout:**
Dropout is a regularization technique where randomly selected neurons are ignored during training, which helps prevent overfitting. The third model, which includes dropout, shows a slight decrease in accuracy compared to the base model but better performance compared to the second model with only L2 regularization.

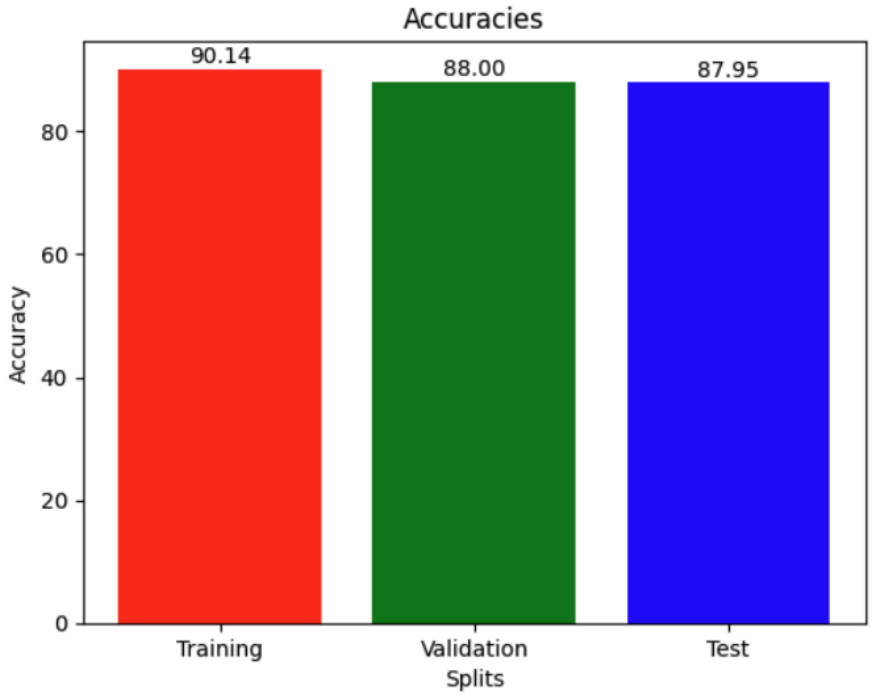
**Early Stopping:**
Early stopping stops training when performance on a validation loss decreases, thereby preventing overfitting. The fourth model, which includes early stopping, shows a decrease in performance compared to the base model and the third model with dropout. This could be because early stopping might have stopped training too early before the model could fully converge which eventually led to the model having suboptimal performance.

3. Discuss the results and provide relevant graphs:

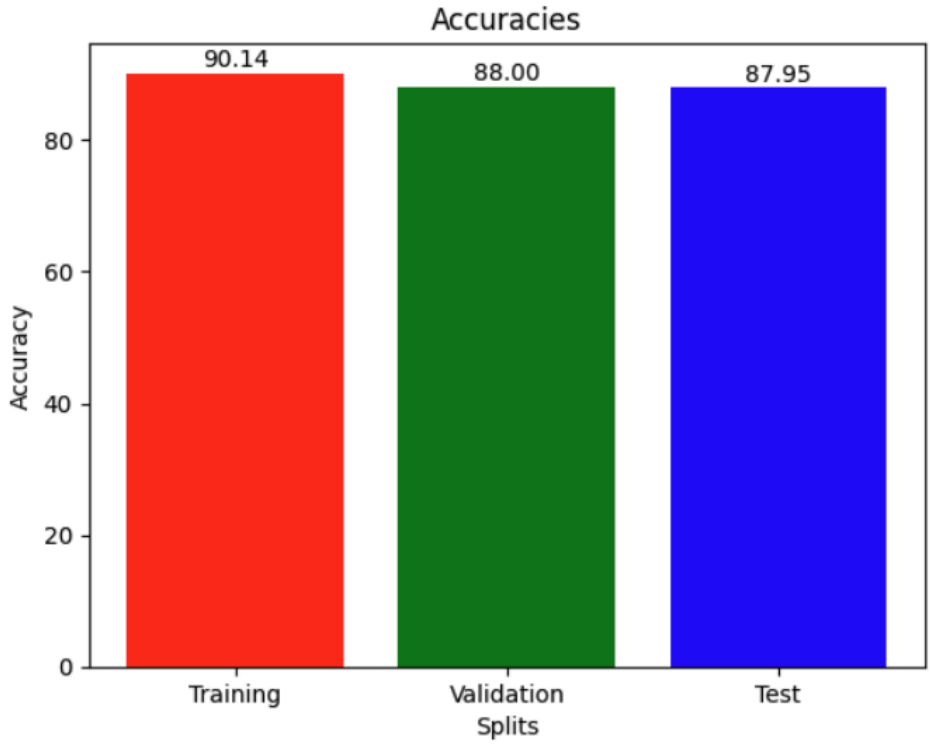
## Base Model

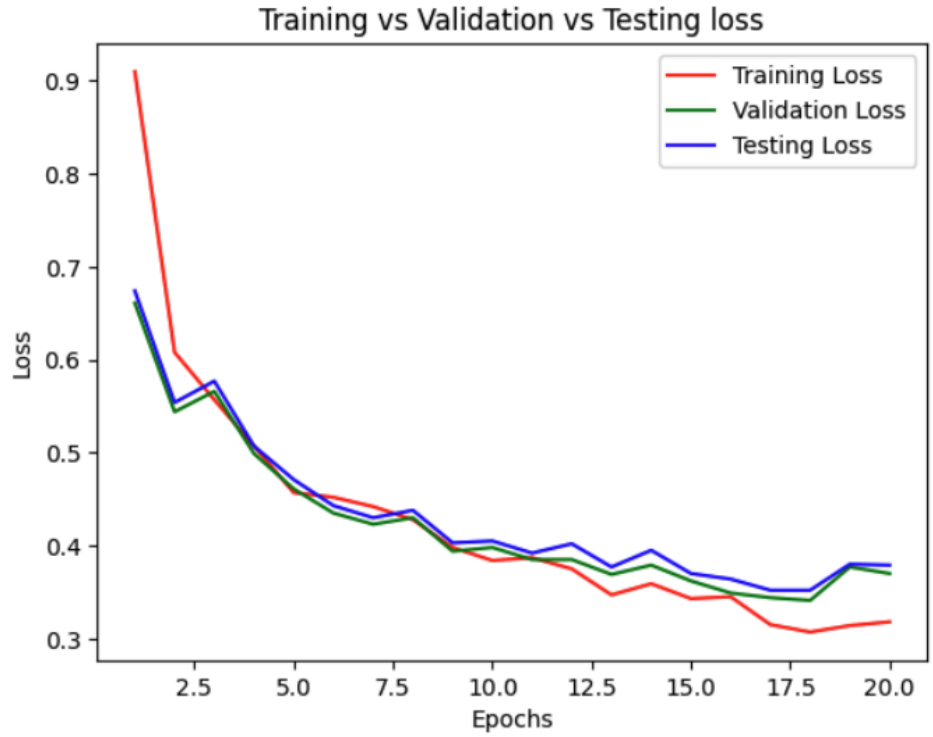
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

```
[1, 500], Training loss: 0.91, Validation loss: 0.661, Testing loss: 0.674
[1, 1000], Training loss: 0.608, Validation loss: 0.544, Testing loss: 0.554
[1, 1500], Training loss: 0.557, Validation loss: 0.566, Testing loss: 0.577
[1, 2000], Training loss: 0.507, Validation loss: 0.499, Testing loss: 0.507
[2, 500], Training loss: 0.457, Validation loss: 0.461, Testing loss: 0.471
[2, 1000], Training loss: 0.452, Validation loss: 0.435, Testing loss: 0.443
[2, 1500], Training loss: 0.442, Validation loss: 0.423, Testing loss: 0.43
[2, 2000], Training loss: 0.428, Validation loss: 0.43, Testing loss: 0.438
[3, 500], Training loss: 0.398, Validation loss: 0.394, Testing loss: 0.403
[3, 1000], Training loss: 0.384, Validation loss: 0.398, Testing loss: 0.405
[3, 1500], Training loss: 0.387, Validation loss: 0.385, Testing loss: 0.392
[3, 2000], Training loss: 0.375, Validation loss: 0.385, Testing loss: 0.402
[4, 500], Training loss: 0.347, Validation loss: 0.369, Testing loss: 0.377
[4, 1000], Training loss: 0.359, Validation loss: 0.379, Testing loss: 0.395
[4, 1500], Training loss: 0.343, Validation loss: 0.362, Testing loss: 0.37
[4, 2000], Training loss: 0.345, Validation loss: 0.349, Testing loss: 0.364
[5, 500], Training loss: 0.315, Validation loss: 0.344, Testing loss: 0.352
[5, 1000], Training loss: 0.307, Validation loss: 0.341, Testing loss: 0.352
[5, 1500], Training loss: 0.314, Validation loss: 0.377, Testing loss: 0.38
[5, 2000], Training loss: 0.318, Validation loss: 0.37, Testing loss: 0.379
Finished Training

Time to train: 12min 31sec
```

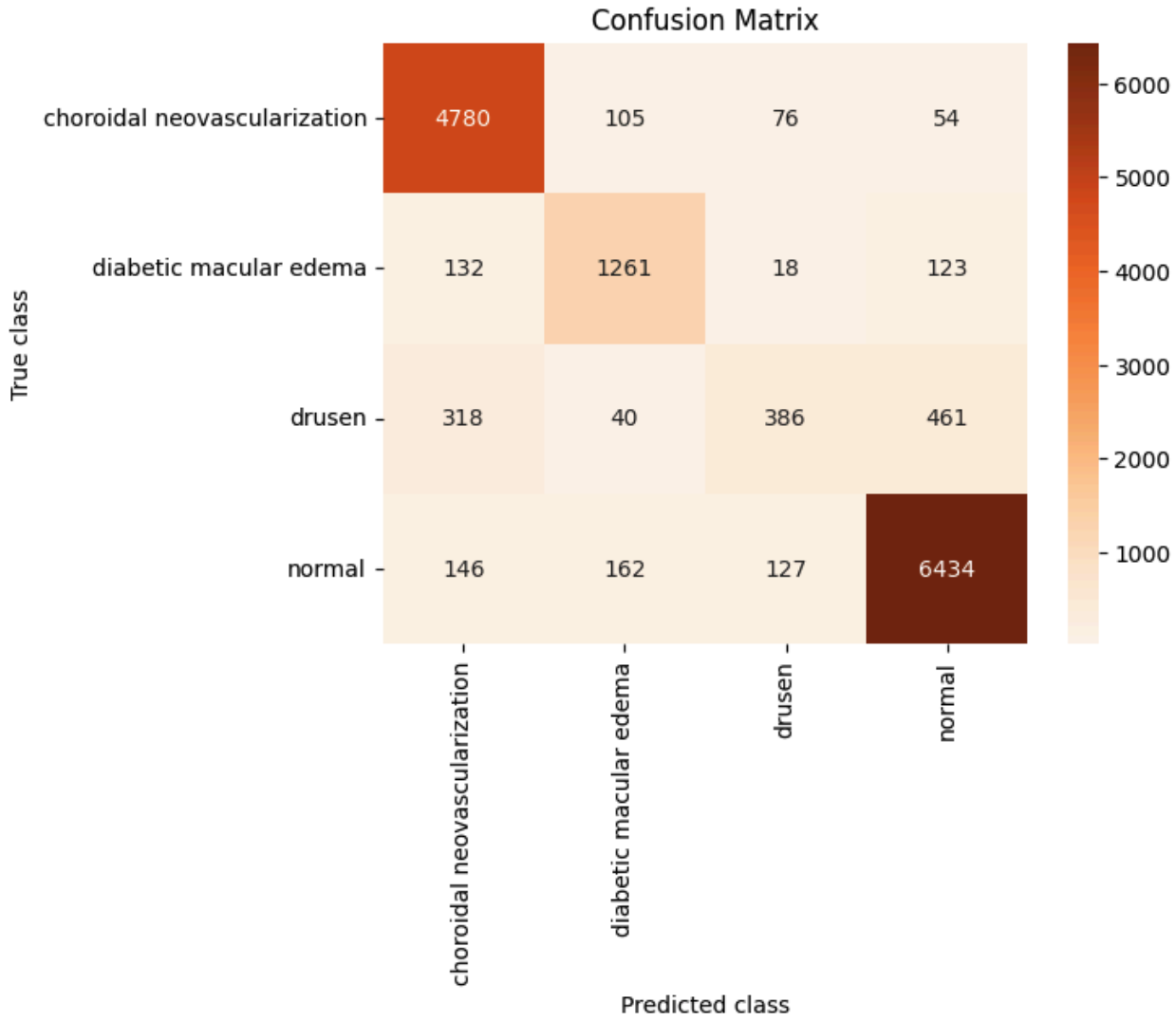


b and c. Plot the training and validation accuracy and loss over time (epochs).



d. Generate a confusion matrix using the model's predictions on the test set.



e. Report any other evaluation metrics used to analyze the model's performance on the test set.

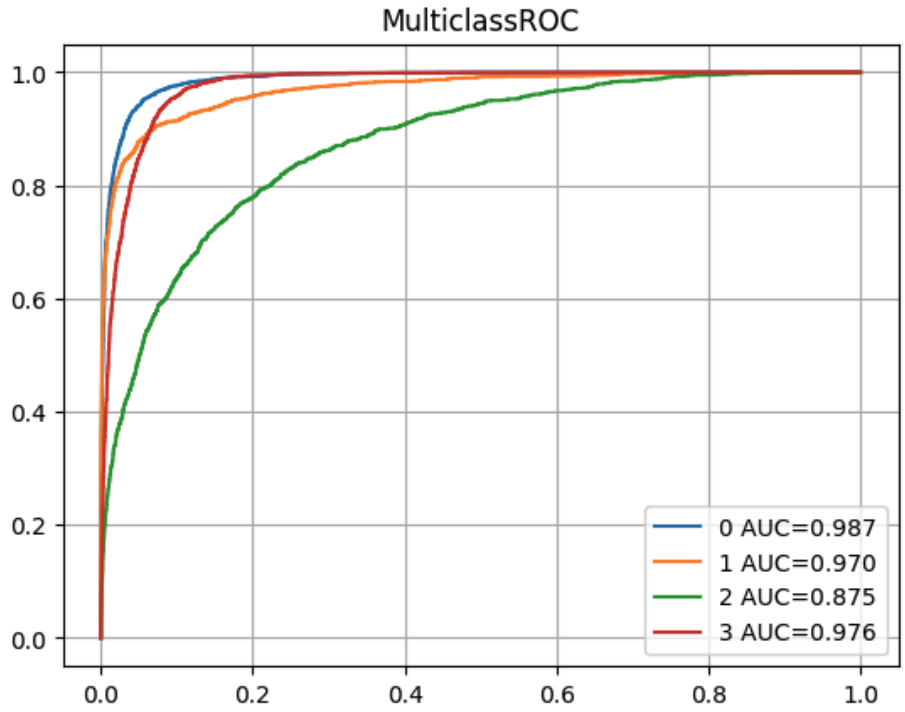
Accuracy, Precision, Recall, F-score:

```
Accuracy on the testing dataset = 87.95
Precision = 0.87
Recall = 0.88
Fscore = 0.87
```

Training Time:

```
Time to train: 12min 31sec
```

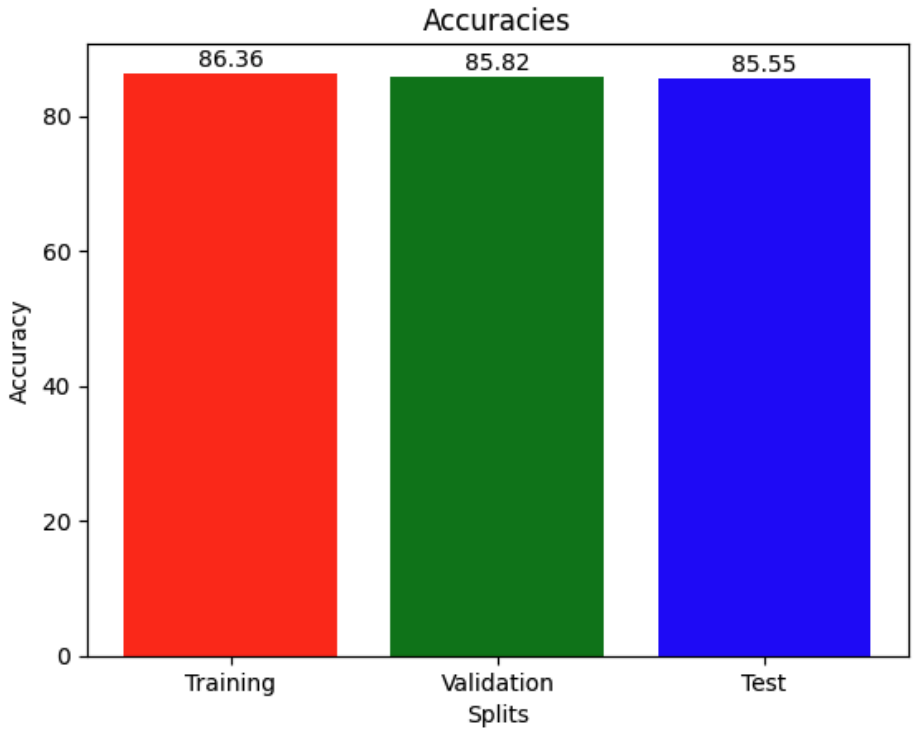
ROC Curve:



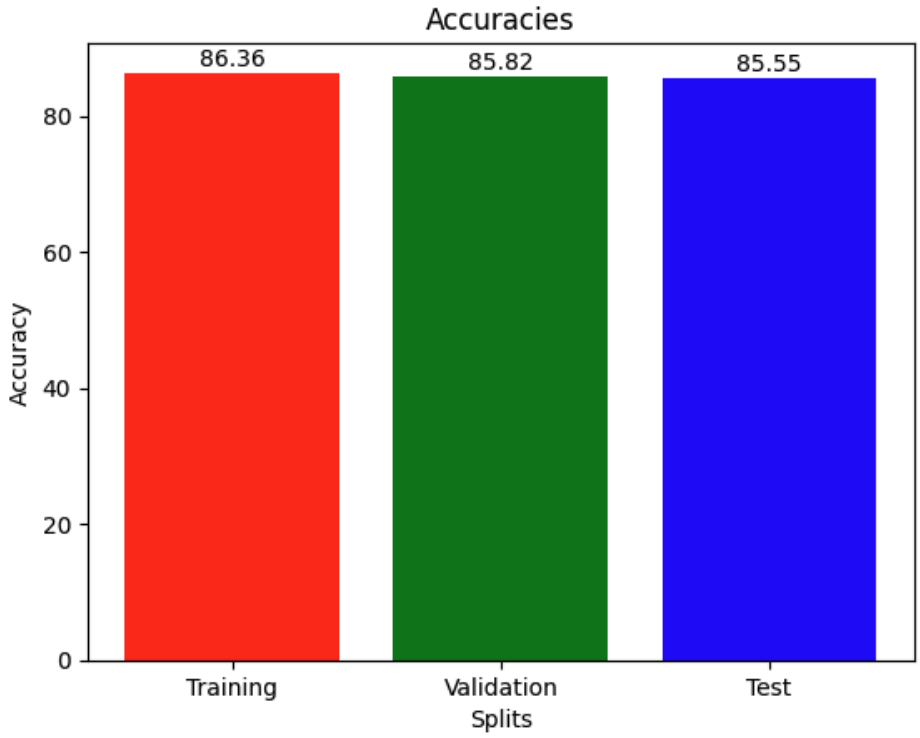
## Model with Regularization

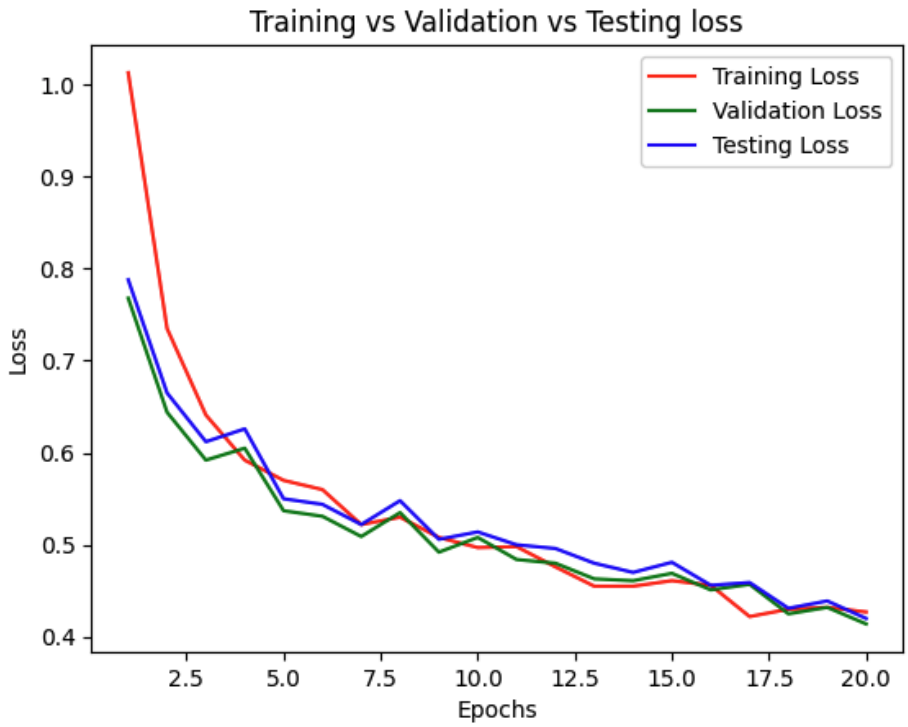
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

```
[1, 500], Training loss: 1.013, Validation loss: 0.768, Testing loss: 0.788
[1, 1000], Training loss: 0.735, Validation loss: 0.644, Testing loss: 0.665
[1, 1500], Training loss: 0.641, Validation loss: 0.592, Testing loss: 0.612
[1, 2000], Training loss: 0.592, Validation loss: 0.605, Testing loss: 0.626
[2, 500], Training loss: 0.57, Validation loss: 0.537, Testing loss: 0.55
[2, 1000], Training loss: 0.56, Validation loss: 0.531, Testing loss: 0.544
[2, 1500], Training loss: 0.522, Validation loss: 0.509, Testing loss: 0.522
[2, 2000], Training loss: 0.53, Validation loss: 0.535, Testing loss: 0.548
[3, 500], Training loss: 0.508, Validation loss: 0.492, Testing loss: 0.506
[3, 1000], Training loss: 0.497, Validation loss: 0.508, Testing loss: 0.514
[3, 1500], Training loss: 0.498, Validation loss: 0.484, Testing loss: 0.5
[3, 2000], Training loss: 0.476, Validation loss: 0.48, Testing loss: 0.496
[4, 500], Training loss: 0.455, Validation loss: 0.463, Testing loss: 0.48
[4, 1000], Training loss: 0.455, Validation loss: 0.461, Testing loss: 0.47
[4, 1500], Training loss: 0.461, Validation loss: 0.469, Testing loss: 0.481
[4, 2000], Training loss: 0.456, Validation loss: 0.451, Testing loss: 0.456
[5, 500], Training loss: 0.422, Validation loss: 0.457, Testing loss: 0.459
[5, 1000], Training loss: 0.43, Validation loss: 0.425, Testing loss: 0.431
[5, 1500], Training loss: 0.432, Validation loss: 0.432, Testing loss: 0.439
[5, 2000], Training loss: 0.427, Validation loss: 0.414, Testing loss: 0.42
Finished Training

Time to train: 12min 10sec
```

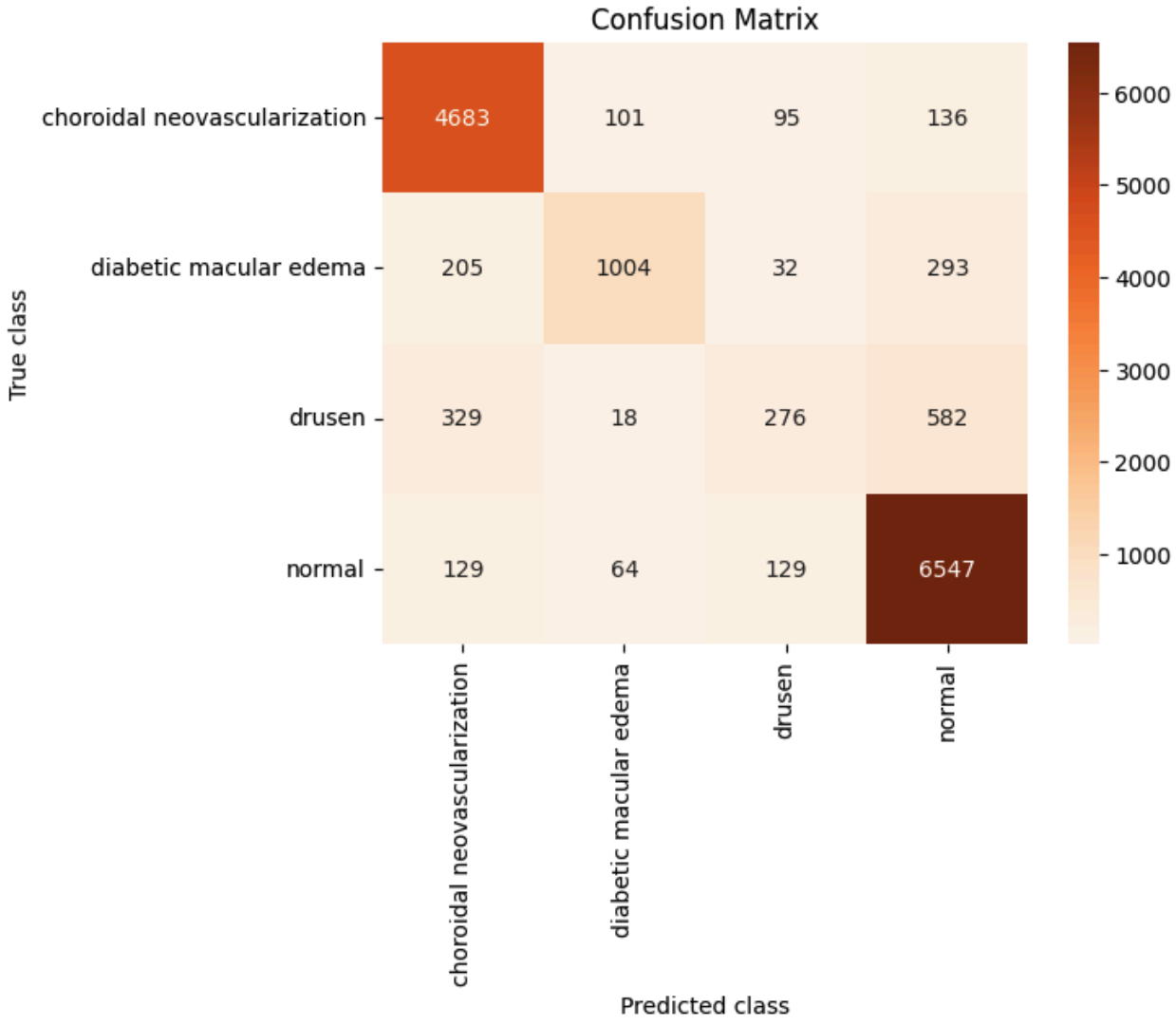


b and c. Plot the training and validation accuracy and loss over time (epochs).



d. Generate a confusion matrix using the model's predictions on the test set.



e. Report any other evaluation metrics used to analyze the model's performance on the test set.

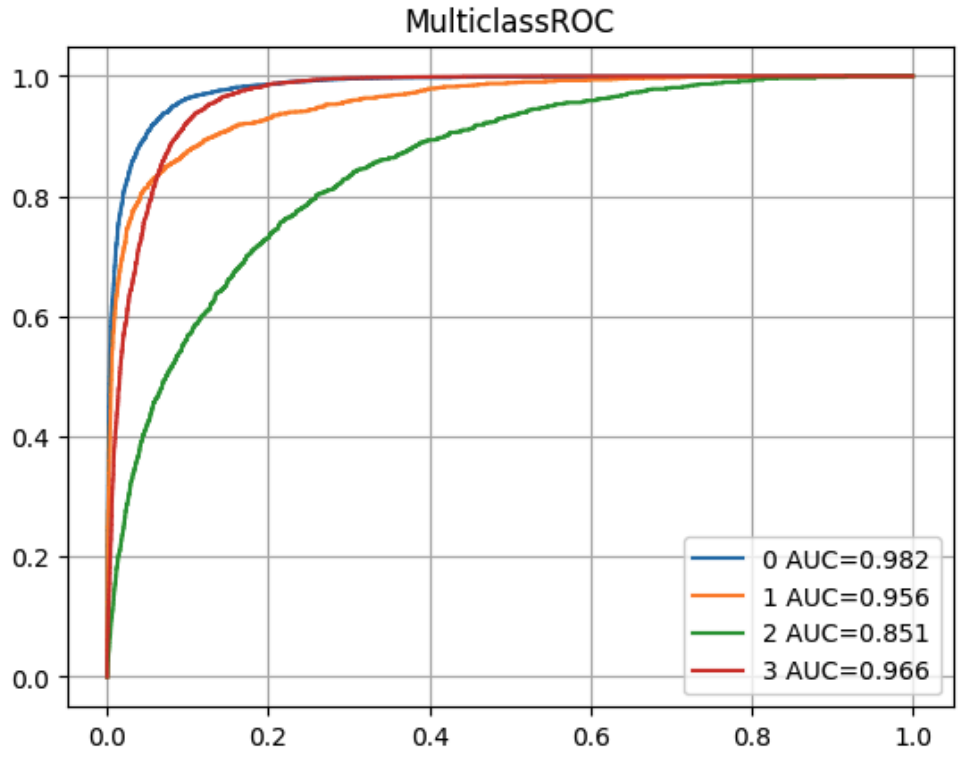
Accuracy, Precision, Recall, F-score:

```
Accuracy on the testing dataset = 85.55
Precision = 0.84
Recall = 0.86
Fscore = 0.84
```

Training Time:

```
Time to train: 12min 10sec
```

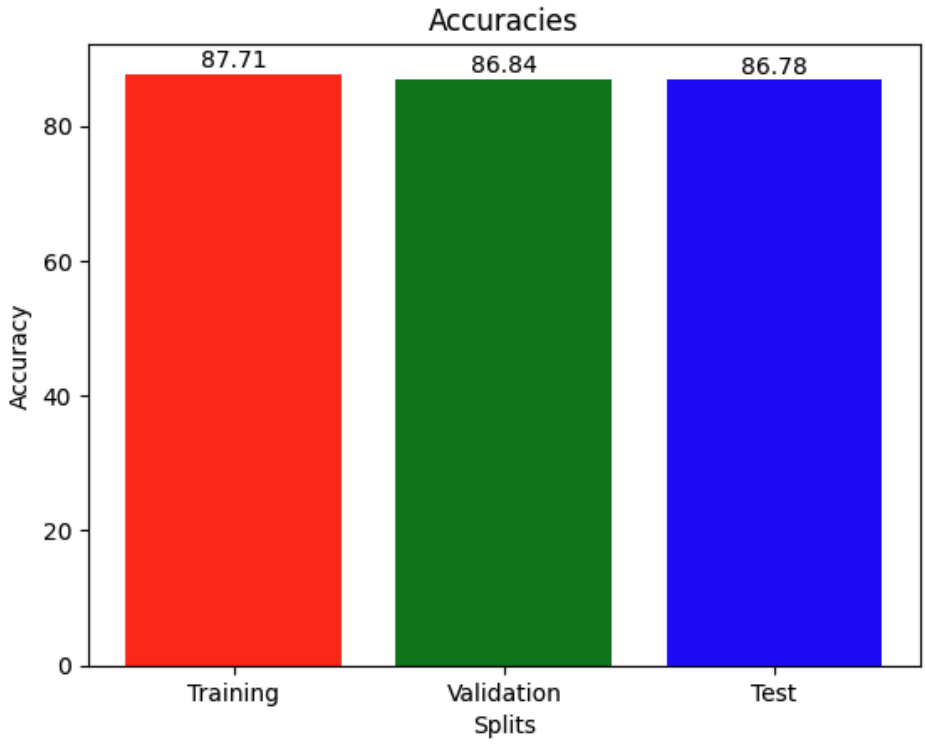
ROC Curve:



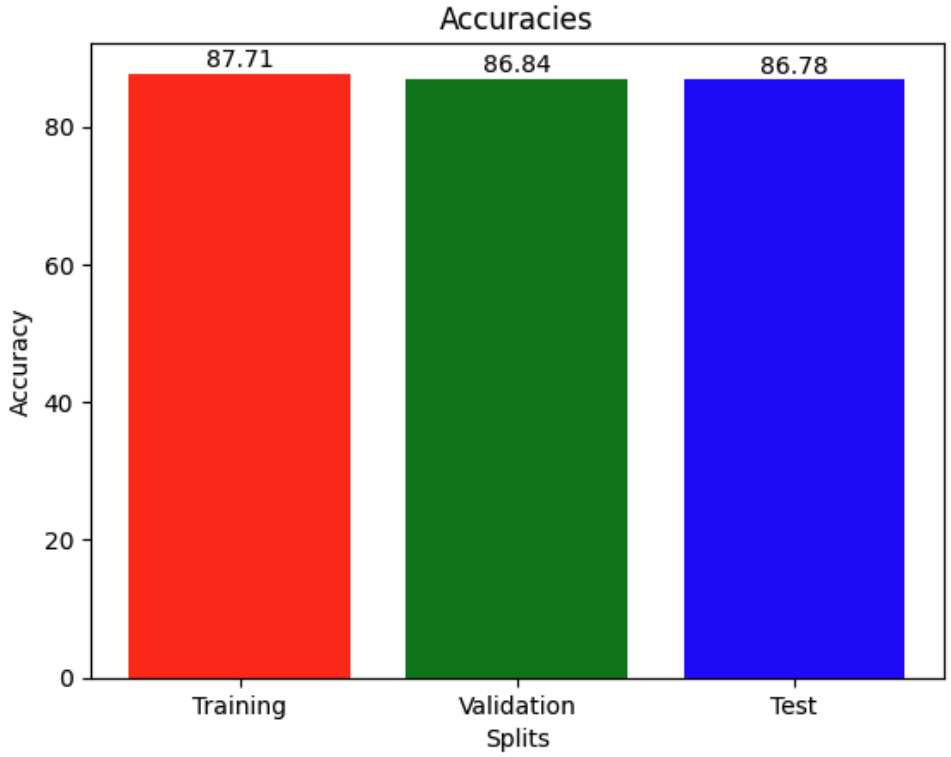
## Model with Regularization and Dropout

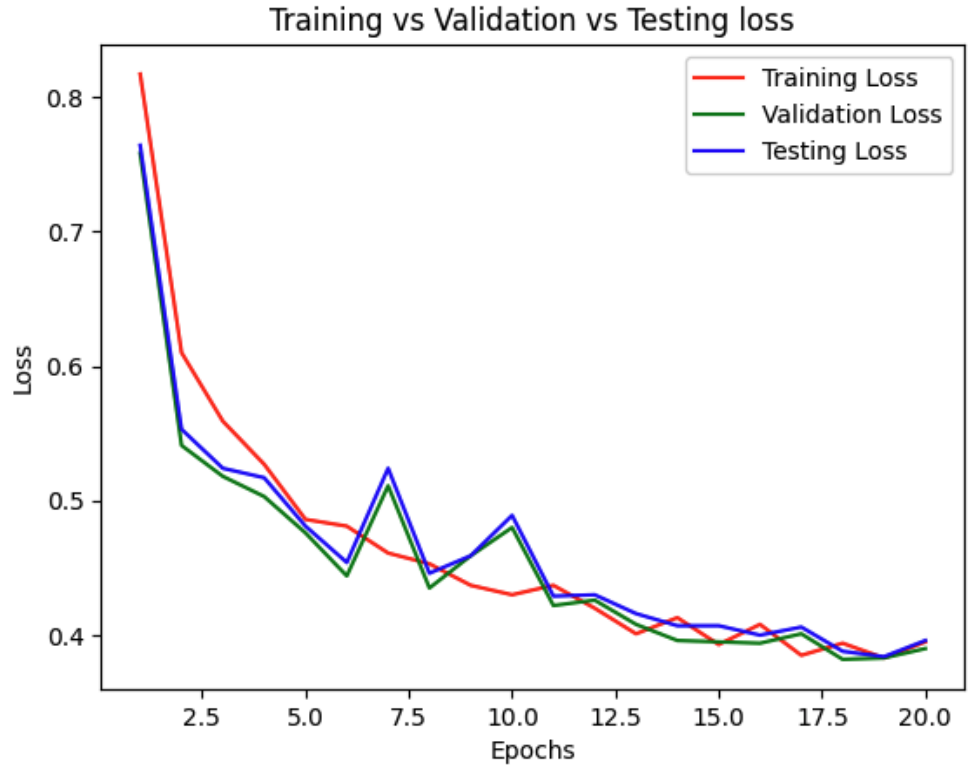
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

```
[1, 500], Training loss: 0.817, Validation loss: 0.758, Testing loss: 0.764
[1, 1000], Training loss: 0.61, Validation loss: 0.541, Testing loss: 0.553
[1, 1500], Training loss: 0.559, Validation loss: 0.518, Testing loss: 0.524
[1, 2000], Training loss: 0.527, Validation loss: 0.503, Testing loss: 0.517
[2, 500], Training loss: 0.486, Validation loss: 0.476, Testing loss: 0.481
[2, 1000], Training loss: 0.481, Validation loss: 0.444, Testing loss: 0.454
[2, 1500], Training loss: 0.461, Validation loss: 0.511, Testing loss: 0.524
[2, 2000], Training loss: 0.453, Validation loss: 0.435, Testing loss: 0.446
[3, 500], Training loss: 0.437, Validation loss: 0.459, Testing loss: 0.459
[3, 1000], Training loss: 0.43, Validation loss: 0.48, Testing loss: 0.489
[3, 1500], Training loss: 0.437, Validation loss: 0.422, Testing loss: 0.429
[3, 2000], Training loss: 0.42, Validation loss: 0.426, Testing loss: 0.43
[4, 500], Training loss: 0.401, Validation loss: 0.408, Testing loss: 0.416
[4, 1000], Training loss: 0.413, Validation loss: 0.396, Testing loss: 0.407
[4, 1500], Training loss: 0.393, Validation loss: 0.395, Testing loss: 0.407
[4, 2000], Training loss: 0.408, Validation loss: 0.394, Testing loss: 0.4
[5, 500], Training loss: 0.385, Validation loss: 0.401, Testing loss: 0.406
[5, 1000], Training loss: 0.394, Validation loss: 0.382, Testing loss: 0.388
[5, 1500], Training loss: 0.383, Validation loss: 0.383, Testing loss: 0.384
[5, 2000], Training loss: 0.395, Validation loss: 0.39, Testing loss: 0.396
Finished Training

Time to train: 12min 8sec
```

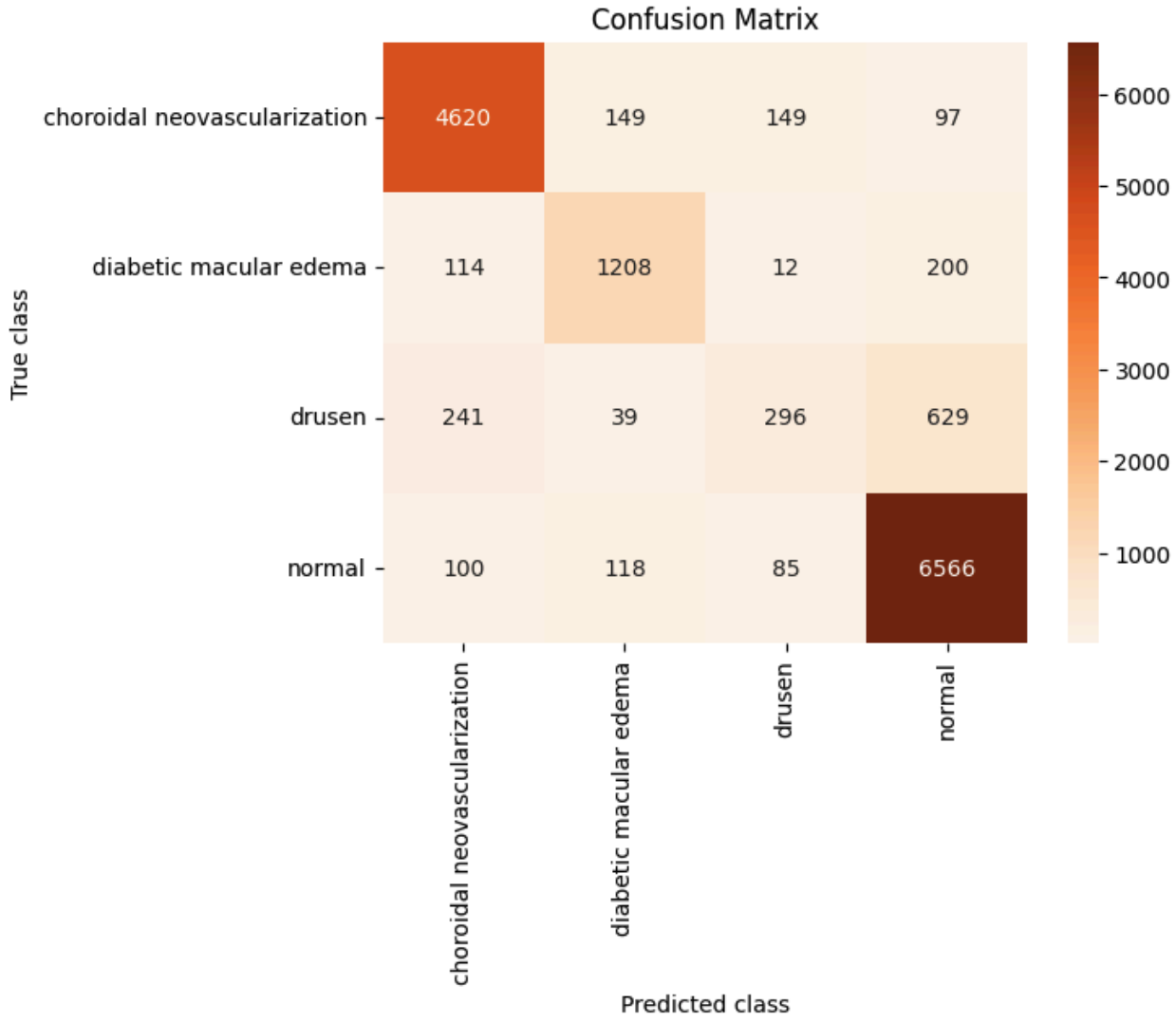


b and c. Plot the training and validation accuracy and loss over time (epochs).



d. Generate a confusion matrix using the model's predictions on the test set.



e. Report any other evaluation metrics used to analyze the model's performance on the test set.

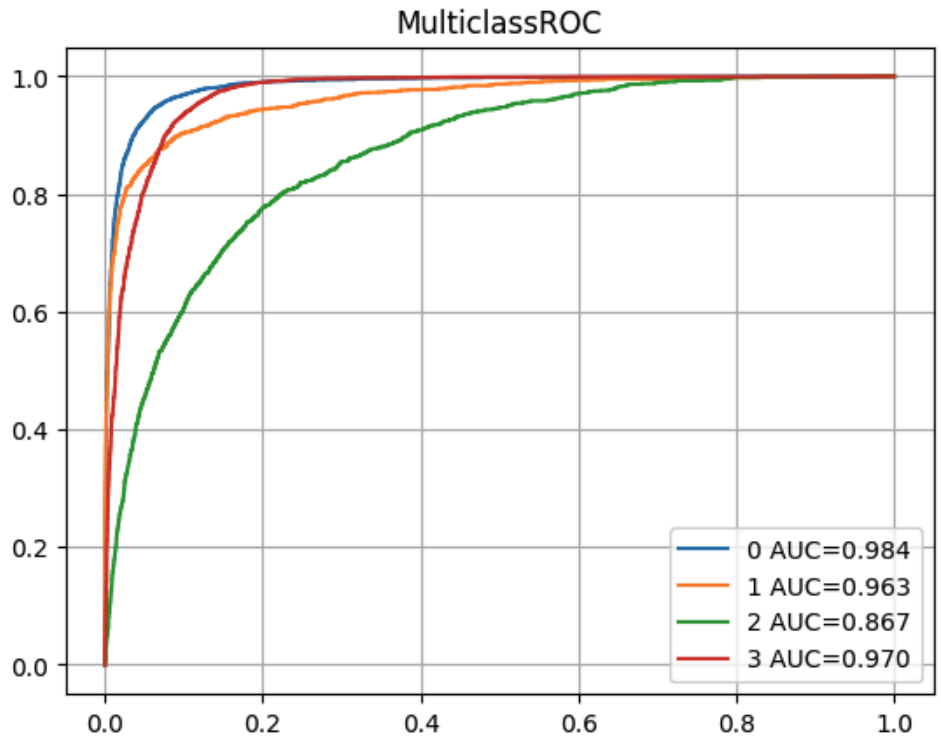
Accuracy, Precision, Recall, F-score:

```
Accuracy on the testing dataset = 86.78
Precision = 0.85
Recall = 0.87
Fscore = 0.85
```

Training Time:

```
Time to train: 12min 8sec
```

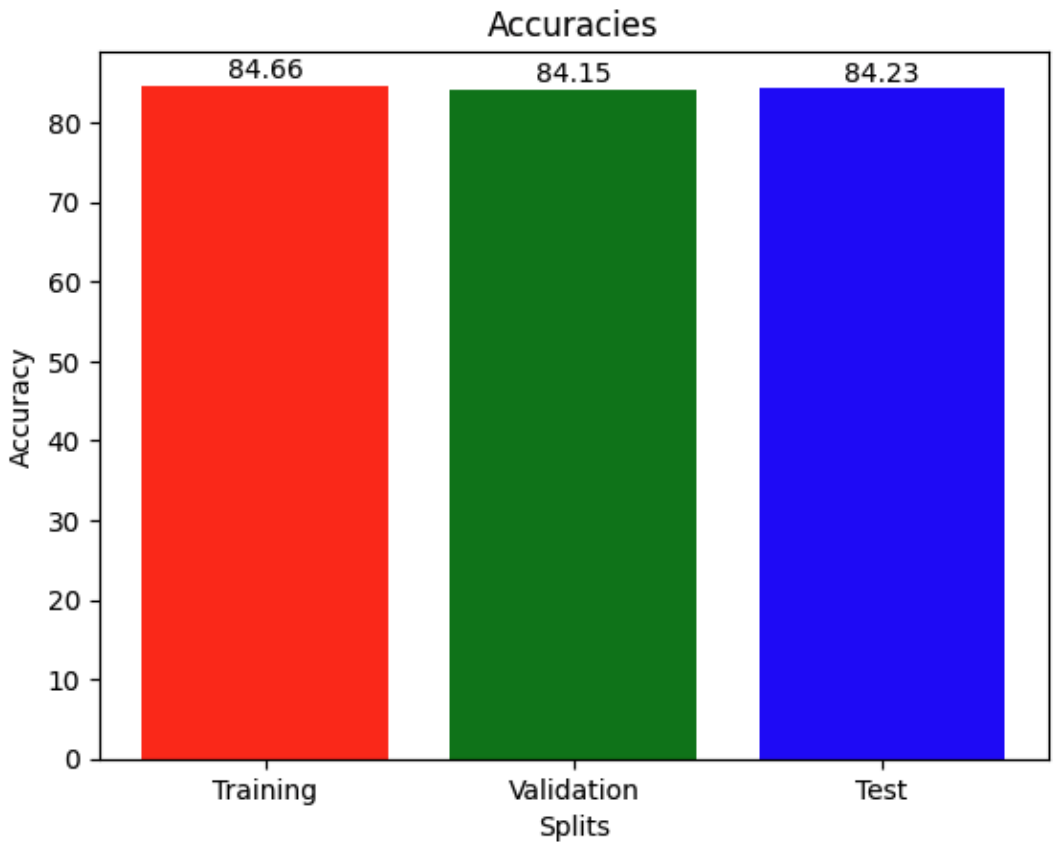
ROC Curve:



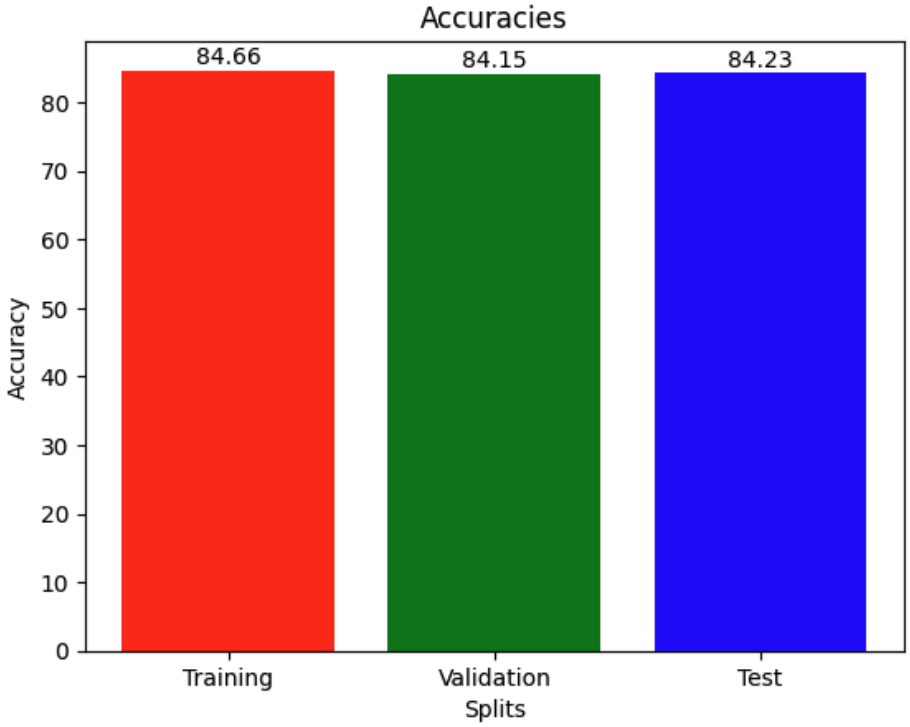
## Model with Regularization, Dropout and Early Stopping

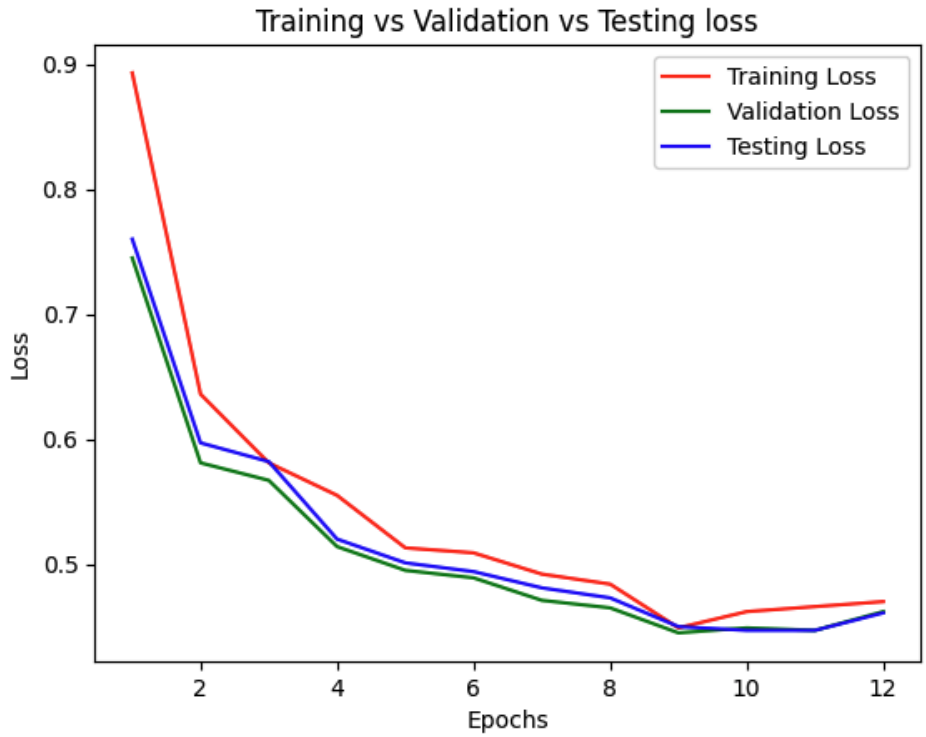
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

```
[1, 500], Training loss: 0.893, Validation loss: 0.745, Testing loss: 0.76
[1, 1000], Training loss: 0.636, Validation loss: 0.581, Testing loss: 0.597
[1, 1500], Training loss: 0.581, Validation loss: 0.567, Testing loss: 0.582
[1, 2000], Training loss: 0.555, Validation loss: 0.514, Testing loss: 0.52
[2, 500], Training loss: 0.513, Validation loss: 0.495, Testing loss: 0.501
[2, 1000], Training loss: 0.509, Validation loss: 0.489, Testing loss: 0.494
[2, 1500], Training loss: 0.492, Validation loss: 0.471, Testing loss: 0.481
[2, 2000], Training loss: 0.484, Validation loss: 0.465, Testing loss: 0.473
[3, 500], Training loss: 0.449, Validation loss: 0.445, Testing loss: 0.45
[3, 1000], Training loss: 0.462, Validation loss: 0.449, Testing loss: 0.447
[3, 1500], Training loss: 0.466, Validation loss: 0.447, Testing loss: 0.447
[3, 2000], Training loss: 0.47, Validation loss: 0.462, Testing loss: 0.461

Stopping training as Validation Loss stopped improving.
Finished Training

Time to train: 7min 17sec
```

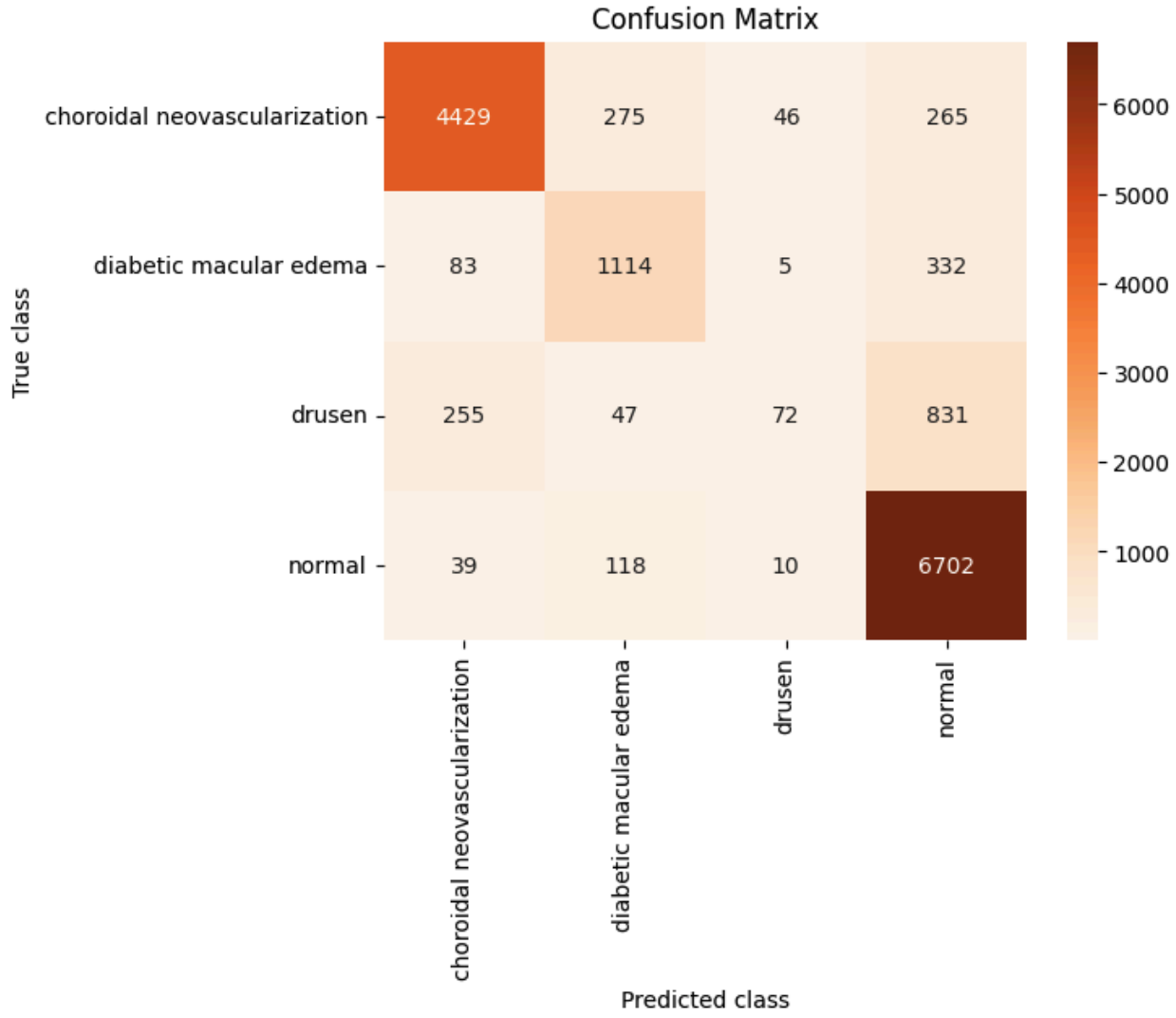


b and c. Plot the training and validation accuracy and loss over time (epochs).



d. Generate a confusion matrix using the model's predictions on the test set.



e. Report any other evaluation metrics used to analyze the model's performance on the test set.

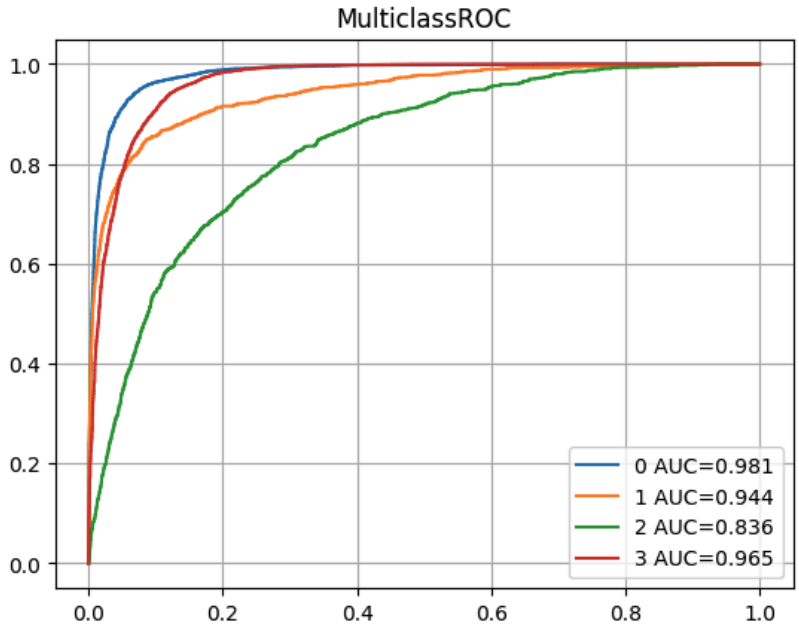
Accuracy, Precision, Recall, F-score:

```
Accuracy on the testing dataset = 84.23
Precision = 0.82
Recall = 0.84
Fscore = 0.81
```

Training Time:

```
Time to train: 7min 17sec
```

ROC Curve:




**Conclusion:** The base model gave the best accuracy among all the models with an accuracy of 87.95 on the testing dataset. On the other hand fourth model with early stopping took significantly less time to train i.e. 7min 17sec

# Bonus points

## Deploy the model

## Model Deployment Video Link

https://drive.google.com/file/d/1lVr7lc7DLB0nkr-CBR9UuwPDZQwtNKWH/view?usp=sharing

## References:


1. https://pandas.pydata.org/docs/
2. https://numpy.org/doc/
3. https://matplotlib.org/stable/index.html
4. https://scikit-learn.org/stable/
5. https://seaborn.pydata.org/
6. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
7. https://optuna.org/
8. https://numpy.org/doc/stable/
9. https://flask.palletsprojects.com/en/3.0.x/
10. Part I - Step 3 is based on CSE 574 Machine Learning Quiz 5 submission by Nikhil Gupta