(Group_10) Assignment_4

Prepared for

Prof: Murat SIMSEK
TA: Samhita Kuili

Faculty of Engineering, University of u Ottawa

ELG5255: Applied Machine Learning

Prepared by

Yousef Abd Al Haleem Ahmed Shindy

Esraa Mahmoud Said Ahmed

Esmael Alahmady Ebrahim Ezz

$Part\ 1$

$a)$

$$GINI(t) = 1 - \sum_j [\ P(j \mid t)\ ]^2$$

$$GINI_{split} = 1 - \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

$Weather:$

	Cloudy	Sunny	Rainy
Yes	2	0	2
No	1	4	1

$GINI(Cloudy) = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = \frac{4}{9}$
$GINI(Sunny) = 1 - (\frac{4}{4})^2 = 0$
$GINI(Rainy) = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = \frac{4}{9}$
$GINI(Weather) = (\frac{3}{10} * \frac{4}{9}) + (\frac{3}{10} * \frac{4}{9}) \approx 0.266$

$Temperature:$

	Hot	Mild	Cold	Cool
Yes	1	2	1	0
No	3	2	0	1

$GINI(Hot) = 1 - (\frac{1}{4})^2 - (\frac{3}{4})^2 = \frac{3}{8}$
$GINI(Mild) = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 = \frac{1}{2}$
$GINI(Cold) = 1 - (\frac{1}{1})^2 = 0$
$GINI(Cool) = 1 - (\frac{1}{1})^2 = 0$
$GINI(Temperature) = (\frac{3}{8} * \frac{4}{10}) + (\frac{1}{2} * \frac{4}{10}) \approx 0.35$

$Humidity:$

	High	Normal
Yes	1	3
No	5	1

$GINI(High) = 1 - (\frac{5}{6})^2 - (\frac{1}{6})^2 = \frac{5}{18}$
$GINI(Normal) = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = \frac{3}{8}$
$GINI(Humidity) = (\frac{6}{10} * \frac{5}{18}) + (\frac{4}{10} * \frac{3}{8}) \approx 0.317$

$Wind:$

	Strong	Weak
Yes	2	2
No	5	1

$GINI(Strong) = 1 - (\frac{2}{7})^2 - (\frac{5}{7})^2 = \frac{20}{49}$
$GINI(Weak) = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = \frac{4}{9}$
$GINI(Wind) = (\frac{7}{10} * \frac{20}{49}) + (\frac{3}{10} * \frac{4}{9}) \approx 0.42$
The smallest GINI is Weather so we divide first based on it.

$Temperature:$

	Hot	Mild	Cold	Cool
Yes	1	2	1	0
No	1	0	0	1

$GINI(Hot) = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = \frac{1}{2}$
$GINI(Mild) = 1 - (\frac{2}{2})^2 = 0$
$GINI(Cold) = 1 - (\frac{1}{1})^2 = 0$
$GINI(Cool) = 1 - (\frac{1}{1})^2 = 0$
$GINI(Temperature) = \frac{2}{6} * \frac{1}{2} \approx 0.17$

$Humidity:$

	High	Normal
Yes	1	3
No	1	1

$GINI(High) = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = \frac{1}{2}$
$GINI(Normal) = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = \frac{3}{8}$
$GINI(Humidity) = (\frac{1}{2} * \frac{2}{6}) + (\frac{3}{8} * \frac{4}{6}) \approx 0.417$

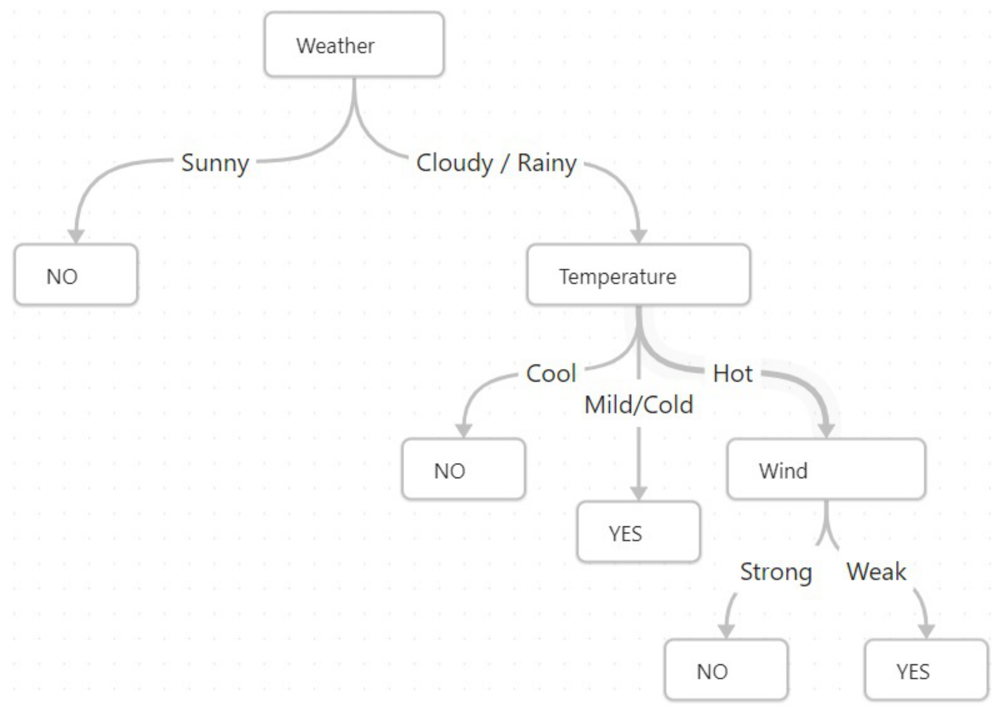
$Wind:$

	Strong	Weak
Yes	2	2
No	2	0

$GINI(Strong) = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 = \frac{1}{2}$
$GINI(Weak) = 1 - (\frac{2}{2})^2 = 0$
$GINI(Wind) = (\frac{1}{2} * \frac{4}{6}) \approx 0.34$

The smallest GINI is Temperature so that's what we divide next based on then we divide either on Humidity or Wind because they both have the same GINI value of 0.

$GINI(Wind) = 1 - (\frac{1}{1})^2 = 0$
$GINI(Humidity) = 1 - (\frac{1}{1})^2 = 0$



$b)$

$$Entropy(t) = -\sum_{j} P(\frac{j}{t}) * log\ p(\frac{j}{t})$$

$$GAIN_{split} = Entropy(p) - [-\sum_{i=1}^{k} \frac{n_i}{n} * Entropy(i)\ ]$$

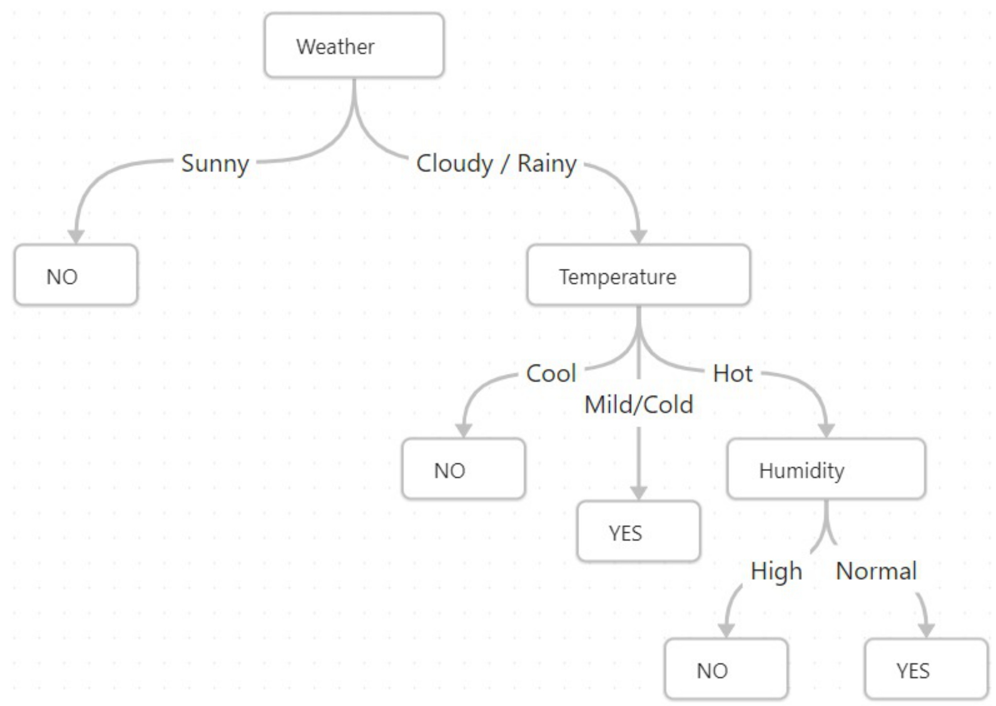
$Entropy(Hiking) = -\frac{4}{10} log_2 \frac{4}{10}\ \ -\ \ \frac{6}{10} log_2 \frac{6}{10}\ \approx\ 0.971$
$GAIN(Weather) = 0.971 - \frac{3}{10}[-\frac{2}{3} log_2 \frac{2}{3} - \frac{1}{3} log_2 \frac{1}{3}] - \frac{4}{10}[-\frac{4}{4} log_2 \frac{4}{4}] - \frac{3}{10}[-\frac{2}{3} log_2 \frac{2}{3} - \frac{1}{3} log_2 \frac{1}{3}] \approx 0.421$
$GAIN(Temperature) = 0.971 - \frac{4}{10}[-\frac{3}{4} log_2 \frac{3}{4} - \frac{1}{4} log_2 \frac{1}{4}] - \frac{4}{10}[-\frac{2}{4} log_2 \frac{2}{4} - \frac{2}{4} log_2 \frac{2}{4}] - \frac{1}{10}[-\frac{1}{10} log_2 \frac{1}{10}] - \frac{1}{10}[-\frac{1}{10} log_2 \frac{1}{10}] \approx 0.247$
$GAIN(Humidity) = 0.971 - \frac{6}{10}[-\frac{5}{6} log_2 \frac{5}{6} - \frac{1}{6} log_2 \frac{1}{6}] - \frac{4}{10}[-\frac{3}{4} log_2 \frac{3}{4} - \frac{1}{4} log_2 \frac{1}{4}] \approx 0.257$
$GAIN(Wind) = 0.971 - \frac{7}{10}[-\frac{5}{7} log_2 \frac{5}{7} - \frac{2}{7} log_2 \frac{2}{7}] - \frac{3}{10}[-\frac{2}{3} log_2 \frac{2}{3} - \frac{1}{3} log_2 \frac{1}{3}] = 0.092 \approx 0.092$

The highest Gain is GAIN(Weather) so that's what we divide first based on.

$Entropy(Hiking) = -\frac{2}{6} log_2 \frac{2}{6} - \frac{4}{6} log_2 \frac{4}{6}\ \approx\ 0.918$
$GAIN(Temperature) = 0.918 - \frac{2}{6}[-\frac{1}{2} log_2 \frac{1}{2} - \frac{1}{2} log_2 \frac{1}{2}] - \frac{1}{6}[-\frac{1}{1} log_2 \frac{1}{1}] - \frac{2}{6}[-\frac{2}{2} log_2 \frac{2}{2}] - \frac{1}{6}[-\frac{1}{1} log_2 \frac{1}{1}] \approx 0.585$
$GAIN(Humidity) = 0.918 - \frac{2}{6}[-\frac{1}{2} log_2 \frac{1}{2} - \frac{1}{2} log_2 \frac{1}{2}] - \frac{4}{6}[-\frac{3}{4} log_2 \frac{3}{4} - \frac{1}{4} log_2 \frac{1}{4}] \approx 0.043$
$GAIN(Wind) = 0.918 - \frac{4}{6}[-\frac{2}{4} log_2 \frac{2}{4} - \frac{2}{4} log_2 \frac{2}{4}] - \frac{2}{6}[-\frac{2}{2} log_2 \frac{2}{2}] \approx 0.251$

The highest Gain is GAIN(Temperature) so that's what we divide second based on and after that we either branch based on Humidity or Wind because they both have the same GAIN value of 1 .

$Entropy(Hiking) = -\frac{1}{2} log_2 \frac{1}{2} - \frac{1}{2} log_2 \frac{1}{2}\ = 1$
$Gain(Wind) = 1 - \frac{1}{2}[-\frac{1}{1} log_2 \frac{1}{1}] - \frac{1}{2}[-\frac{1}{1} log_2 \frac{1}{1}] = 1$
$Gain(Humidity) = 1 - \frac{1}{2}[-\frac{1}{1} log_2 \frac{1}{1}] - \frac{1}{2}[-\frac{1}{1} log_2 \frac{1}{1}] = 1$



$c)$
$GINI:$
Advantages: Simplicity as it is relatively simple and straightforward. It Can handle multi-class problems and their squared proportions. It's Computationally efficient and Suitable for categorical variables and binary classification tasks.

Disadvantages: It ignores actual probabilities, focusing only on class frequencies and is biased towards attributes with many distinct values.

$InformationGain:$
Advantages: Takes into account actual probabilities opposed to Gini index, Works with categorical and numerical attributes and can also handle multi-class problems

Disadvantages: Computationally more expensive as Calculating the Information Gain involves calculating logarithms and conditional probabilities, which can be computationally more expensive compared to Gini Index.

Part 2:

1. **a)** Reading and viewing the dataset with column names from the csv and showing the shape of the dataset (38 input and 1 target).

```
old_df = pd.read_csv("/content/KDD.csv")
old_df.head()
```

dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	target
0.0	0.11	0.0	0.0	0.0	0.0	0.0	0
0.0	0.05	0.0	0.0	0.0	0.0	0.0	0
0.0	0.03	0.0	0.0	0.0	0.0	0.0	0
0.0	0.03	0.0	0.0	0.0	0.0	0.0	0
0.0	0.02	0.0	0.0	0.0	0.0	0.0	0

```
old_df.shape[1]
```

39

Then we use MinMaxScaler to normalize the data

```
scaler = MinMaxScaler()
new_x = scaler.fit_transform(x)
new_x = pd.DataFrame(new_x, columns=x.columns)
```

And select the top 9 features and name the data my_data

```
] K_select = SelectKBest(score_func=f_classif, k=9)
  x_new = K_select.fit_transform(new_x, y)
  top_feats = new_x.columns[K_select.get_support()]
  my_data = pd.concat([new_x[top_feats], y], axis=1)
  my_data.head()
```

And view the first 5 rows

	logged_in	count	srv_count	serror_rate	same_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_same_src_port_rate	dst_host_srv_serror_rate	target
0	1.0	0.015656	0.015656	0.0	1.0	0.0	0.035294	0.11	0.0	0
1	1.0	0.015656	0.015656	0.0	1.0	0.0	0.074510	0.05	0.0	0
2	1.0	0.015656	0.015656	0.0	1.0	0.0	0.113725	0.03	0.0	0
3	1.0	0.011742	0.011742	0.0	1.0	0.0	0.152941	0.03	0.0	0
4	1.0	0.011742	0.011742	0.0	1.0	0.0	0.192157	0.02	0.0	0

b) Splitting the data three times with the specified ratios and and compute the performance of Decision tree and show the classification report for each subset

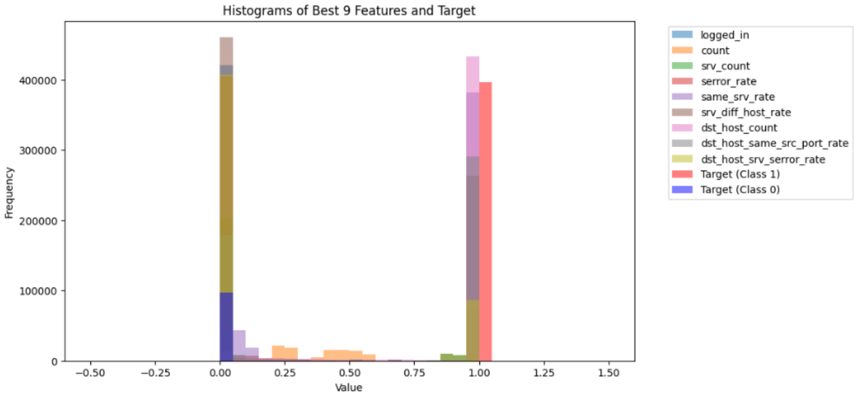
```
split_sizes = [0.3, 0.4, 0.5]#Pre-defined split sizes
for idx,size in enumerate(split_sizes):
    x_train, x_test, y_train, y_test = split_data(x,y,size)
    dtc = DecisionTreeClassifier(random_state=42)
    dtc.fit(x_train, y_train)
    y_pred = dtc.predict(x_test)
    class_rep = classification_report(y_test, y_pred)
    print(f"Classification Report for my_data_{idx+1}:")
    print(class_rep)
    print("---------------------------------------------")
```

```
Classification Report for my_data_1:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     29192
           1       1.00      0.99      0.99    119015

    accuracy                           0.99    148207
   macro avg       0.98      0.99      0.99    148207
weighted avg       0.99      0.99      0.99    148207
```

```
Classification Report for my_data_2:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     38977
           1       1.00      0.99      0.99    158632

    accuracy                           0.99    197609
   macro avg       0.98      0.99      0.99    197609
weighted avg       0.99      0.99      0.99    197609
```

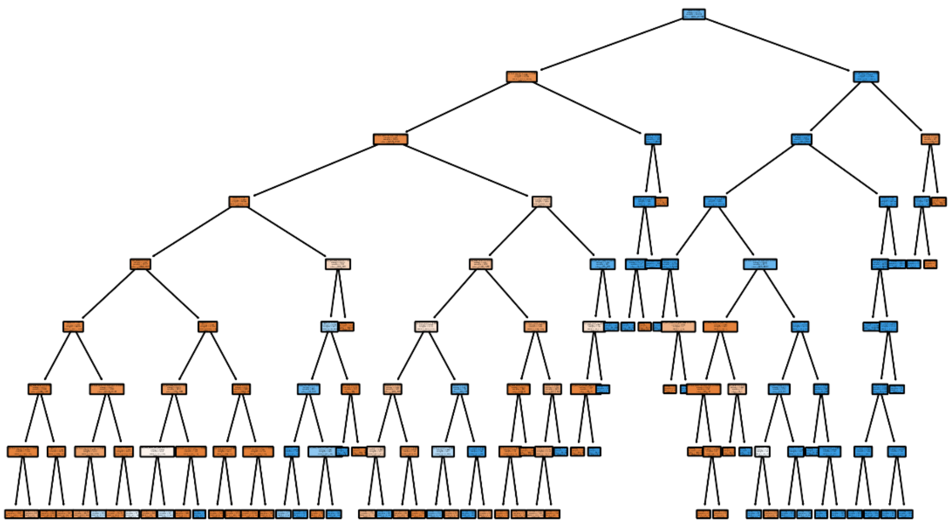
```
Classification Report for my_data_3:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     48650
           1       1.00      0.99      0.99    198361

    accuracy                           0.99    247011
   macro avg       0.98      0.99      0.99    247011
weighted avg       0.99      0.99      0.99    247011
```

and plot a histogram for the top 9 features.

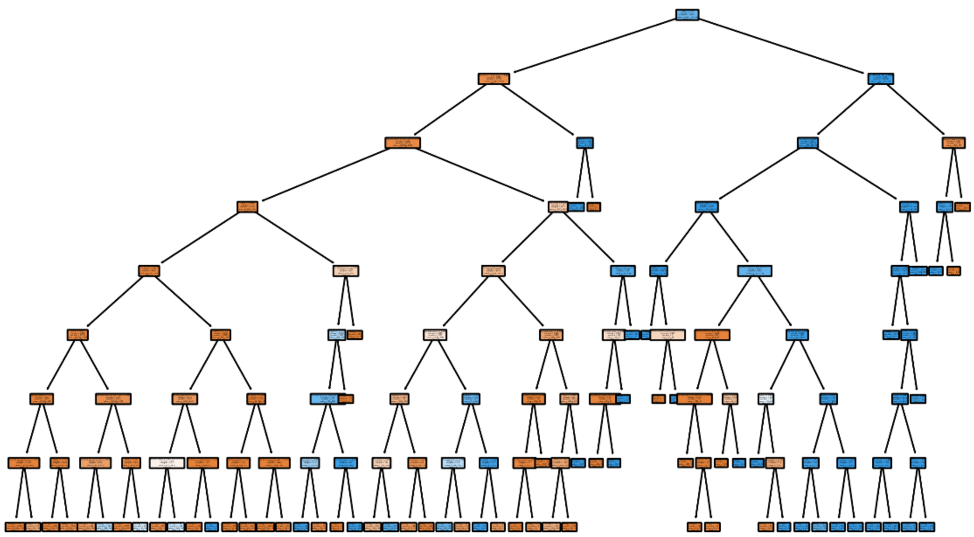


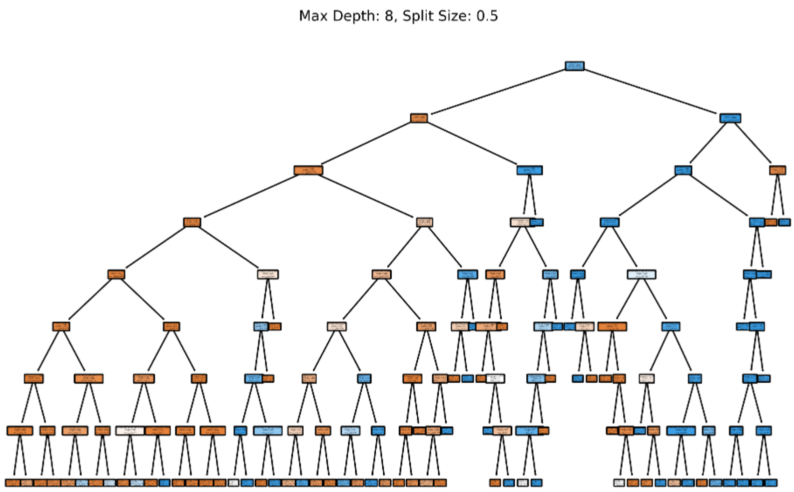
c) Visualizing the best split of decision tree with entropy as a measure of node impurity with the defined **max_depths** and not defining a **max_leaf_nodes** in the model to increases the accuracy. We notice that the highest accuracy for each split size was with **max_depth = 8.**

Max Depth: 8, Split Size: 0.3



Max Depth: 8, Split Size: 0.4

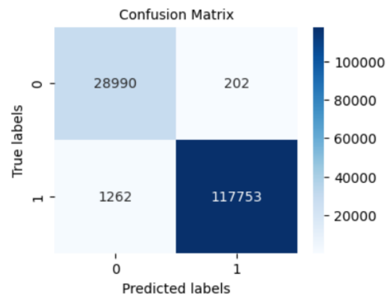




d) Calculate and display the accuracy score, classification report and confusion matrix for tuned decision tree for the three split sizes.

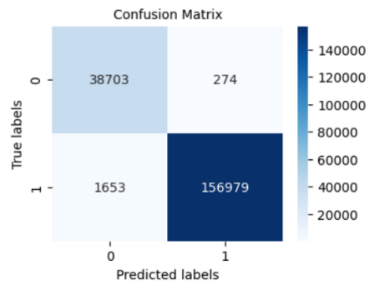
```
Classification Report for tuned model for my_data_1:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     29192
           1       1.00      0.99      0.99    119015

    accuracy                           0.99    148207
   macro avg       0.98      0.99      0.98    148207
weighted avg       0.99      0.99      0.99    148207
```

```
Accuracy: 99.01219240656648
Confusion Matrix:
```



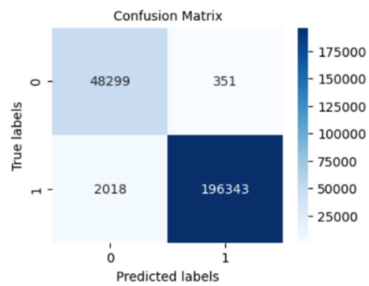

```
Classification Report for tuned model for my_data_2:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     38977
           1       1.00      0.99      0.99    158632

    accuracy                           0.99    197609
   macro avg       0.98      0.99      0.98    197609
weighted avg       0.99      0.99      0.99    197609
```

```
Accuracy: 99.02484198594193
Confusion Matrix:
```

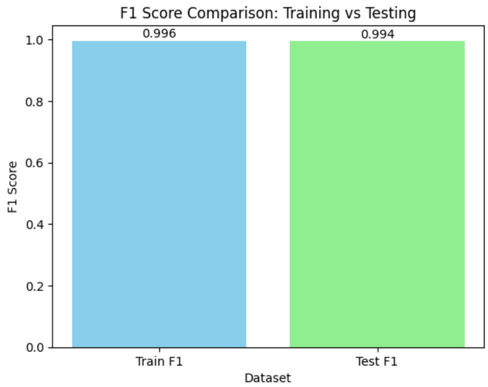



```
Classification Report for tuned model for my_data_3:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     48650
           1       1.00      0.99      0.99    198361

    accuracy                           0.99    247011
   macro avg       0.98      0.99      0.99    247011
weighted avg       0.99      0.99      0.99    247011
```

```
Accuracy: 99.04093339972714
Confusion Matrix:
```



e) Plotting the F1 scores for training and testing of the model and comparing them.



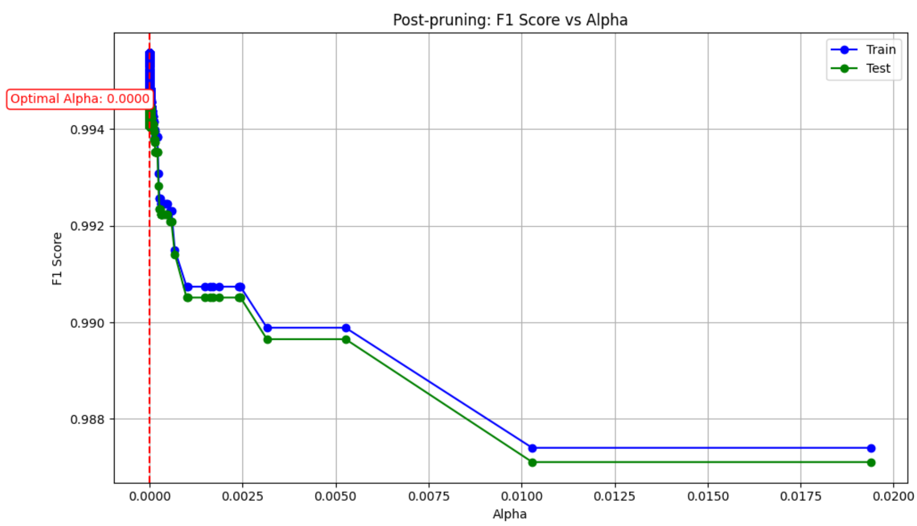
Then we apply Pre-Pruning, Post-Pruning and K-Fold cross validation and compare the resulting F1 scores with the ones we had before.

Pre-Pruning:

```
max_depths = range(1, 11)
train_pre_pruning = []
test_pre_pruning = []
for depth in max_depths:
    clf = DecisionTreeClassifier(criterion='entropy', max_depth=depth, random_state=42)
    clf.fit(x_train, y_train)
    train_pred = clf.predict(x_train)
    test_pred = clf.predict(x_test)

    train_f1 = f1_score(y_train, train_pred)
    test_f1 = f1_score(y_test, test_pred)
    train_pre_pruning.append(train_f1)
    test_pre_pruning.append(test_f1)

fig = go.Figure()
fig.add_trace(go.Scatter(x=list(max_depths), y=train_pre_pruning, mode='lines+markers', name='Train'))
fig.add_trace(go.Scatter(x=list(max_depths), y=test_pre_pruning, mode='lines+markers', name='Test'))
fig.update_layout(
    title='Pre_Pruning',
    xaxis_title='Tree Depth',
    yaxis_title='F1 Score',
    legend_title='Dataset',
    hovermode='x',
    template='plotly_white',
    margin=dict(l=50, r=50, t=80, b=50))
fig.show()
```

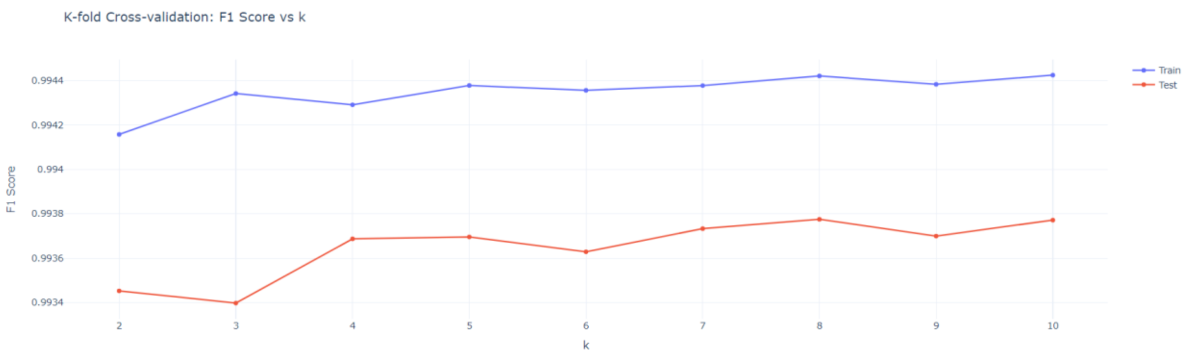
Post-Pruning:

```
dtc = DecisionTreeClassifier(criterion='entropy' ,random_state=42)
dtc.fit(x_train, y_train)
path = dtc.cost_complexity_pruning_path(x_train, y_train)
ccp_alphas = path.ccp_alphas[:-1]
train_post_pruning = []
test_post_pruning = []

for ccp_alpha in ccp_alphas:
    dtc = DecisionTreeClassifier(ccp_alpha=ccp_alpha, random_state=42)
    dtc.fit(x_train, y_train)
    train_pred = dtc.predict(x_train)
    test_pred = dtc.predict(x_test)

    train_f1 = f1_score(y_train, train_pred)
    test_f1 = f1_score(y_test, test_pred)
    train_post_pruning.append(train_f1)
    test_post_pruning.append(test_f1)

optimal_alpha = ccp_alphas[np.argmax(test_post_pruning)]

plt.figure(figsize=(10, 6))
plt.plot(ccp_alphas, train_post_pruning, marker='o', label='Train', linestyle='-', color='blue')
plt.plot(ccp_alphas, test_post_pruning, marker='o', label='Test', linestyle='-', color='green')
plt.xlabel("Alpha")
plt.ylabel("F1 Score")
plt.title("Post-pruning: F1 Score vs Alpha")
plt.grid(True)
plt.tight_layout()
plt.axvline(x=optimal_alpha, color='red', linestyle='--')
plt.text(optimal_alpha, max(test_post_pruning), f'Optimal Alpha: {optimal_alpha:.4f}', color='red',
         verticalalignment='bottom', horizontalalignment='right', fontsize=10, bbox=dict(facecolor='white', edgecolor='red', boxstyle='round'))
plt.legend()
plt.show()
```



K-Fold cross validation:

```python
dtc = DecisionTreeClassifier(criterion='entropy' ,random_state=42)
k_values = range(2, 11)
train_kfold = []
test_kfold = []
for k in k_values:
    train_scores = cross_val_score(dtc, x_train, y_train, cv=k, scoring='f1')
    test_scores = cross_val_score(dtc, x_test, y_test, cv=k, scoring='f1')
    train_kfold.append(np.mean(train_scores))
    test_kfold.append(np.mean(test_scores))

fig = go.Figure()
fig.add_trace(go.Scatter(x=list(k_values), y=train_kfold, mode='lines+markers', name='Train'))
fig.add_trace(go.Scatter(x=list(k_values), y=test_kfold, mode='lines+markers', name='Test'))
fig.update_layout(
    title='K-fold Cross-validation: F1 Score vs k',
    xaxis_title='k',
    yaxis_title='F1 Score',
    hovermode='x',
    template='plotly_white'
)
fig.show()
```



Then we display the train and test F1 scores for before mitigation, Pre-Pruning, Post-Pruning and K-Fold cross validation showing the improvement these techniques made.