```
#%%---------------Q1------------------
import numpy as np
import matplotlib.pyplot as plt

probabilities = np.linspace(0.01,0.99,100)
entropy = -probabilities * np.log2(probabilities) - (1-probabilities) * np.log2(1-
probabilities)
gini_impurity = 2 * (probabilities) * (1-probabilities)
plt.figure(figsize=(8,6))
plt.plot(probabilities,entropy,label='Entropy',linewidth=3)
plt.plot(probabilities,gini_impurity,label='Gini Impurity',linewidth=3)
plt.xlabel("Probability")
plt.ylabel("Entropy/Gini Impurity")
plt.title("Entropy and Gini Impurity vs Probability")
plt.legend()
plt.grid(True)
plt.show()

#%%---------------Q4------------------
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

df = sns.load_dataset('titanic')
df.dropna(how='any', inplace=True)
X=df.select_dtypes(include=['int','float'])
X.drop(columns=['pclass','survived'], inplace=True)
y=df[['survived']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5805, stratify=y)

dt_no_prune = DecisionTreeClassifier(random_state=5805)
dt_no_prune.fit(X_train, y_train)
train_accuracy = accuracy_score(y_train, dt_no_prune.predict(X_train))
test_accuracy = accuracy_score(y_test, dt_no_prune.predict(X_test))
print(f"Training Accuracy (No Pruning): {train_accuracy:.2f}")
print(f"Test Accuracy (No Pruning): {test_accuracy:.2f}")
print("Decision Tree Parameters:", dt_no_prune.get_params())


# Plot the Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(dt_no_prune, feature_names=X.columns, class_names=['Not Survived',
'Survived'], filled=True)
plt.title("Decision Tree for Titanic Dataset (No Pruning)")
plt.show()


#%%---------------Q5------------------
from sklearn.model_selection import GridSearchCV
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score,
```

```
confusion_matrix,roc_curve


df = sns.load_dataset('titanic')
df.dropna(how='any', inplace=True)
X=df.select_dtypes(include=['int','float'])
X.drop(columns=['pclass','survived'], inplace=True)
y=df[['survived']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5805, stratify=y)

tuned_parameters = {
    'max_depth': [1, 2, 3, 4, 5],
    'min_samples_split': [20, 30, 40],
    'min_samples_leaf': [10, 20, 30],
    'criterion': ['gini', 'entropy','log_loss'],
    'splitter': ['best', 'random'],
    'max_features': ['sqrt', 'log2']
}
dt_pre_prune = DecisionTreeClassifier(random_state=5805)
grid_search = GridSearchCV(estimator=dt_pre_prune, param_grid=tuned_parameters,
cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

pre_prune_train_accuracy = accuracy_score(y_train, best_model.predict(X_train))
pre_prune_test_accuracy = accuracy_score(y_test, best_model.predict(X_test))
print(f"Training Accuracy (Pre-pruned Tree): {pre_prune_train_accuracy:.2f}")
print(f"Test Accuracy (Pre-pruned Tree): {pre_prune_test_accuracy :.2f}")

pre_prune_recall = recall_score(y_test, best_model.predict(X_test))
pre_prune_auc = roc_auc_score(y_test, best_model.predict_proba(X_test)[:, 1])
pre_prune_conf_matrix = confusion_matrix(y_test, best_model.predict(X_test))

fpr_pre_prune, tpr_pre_prune, _ = roc_curve(y_test,
best_model.predict_proba(X_test)[:, 1])

plt.figure(figsize=(20, 10))
plot_tree(best_model, feature_names=X.columns, class_names=['Not Survived',
'Survived'], filled=True)
plt.title("Decision Tree for Titanic Dataset (Pre-Pruning)")
plt.show()




#%%---------------Q6------------------
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score,
confusion_matrix, roc_curve
import seaborn as sns
from sklearn.model_selection import train_test_split

df = sns.load_dataset('titanic')
```

```
df.dropna(how='any', inplace=True)
X = df.select_dtypes(include=['int', 'float'])
X.drop(columns=['pclass', 'survived'], inplace=True)
y = df['survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5805, stratify=y)

dt_full = DecisionTreeClassifier(random_state=5805)
dt_full.fit(X_train, y_train)
path = dt_full.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas

train_scores = []
test_scores = []
for alpha in ccp_alphas:
    dt = DecisionTreeClassifier(random_state=5805, ccp_alpha=alpha)
    dt.fit(X_train, y_train)
    train_scores.append(accuracy_score(y_train, dt.predict(X_train)))
    test_scores.append(accuracy_score(y_test, dt.predict(X_test)))

optimal_alpha_index = test_scores.index(max(test_scores))
optimal_alpha = ccp_alphas[optimal_alpha_index]
print("Optimal CCP Alpha:", optimal_alpha)

plt.figure(figsize=(10, 6))
plt.plot(ccp_alphas, train_scores, label='Train Accuracy', marker='o')
plt.plot(ccp_alphas, test_scores, label='Test Accuracy', marker='o')
plt.axvline(x=optimal_alpha, color='red', linestyle='--', label=f'Optimal Alpha =
{optimal_alpha:.4f}')

plt.xlabel("CCP Alpha")
plt.ylabel("Accuracy")
plt.title("Accuracy vs CCP Alpha for Training and Test Sets")
plt.legend()
plt.grid(True)
plt.show()

dt_pruned = DecisionTreeClassifier(random_state=5805, ccp_alpha=optimal_alpha)
dt_pruned.fit(X_train, y_train)
post_prune_train_accuracy = accuracy_score(y_train, dt_pruned.predict(X_train))
post_prune_test_accuracy = accuracy_score(y_test, dt_pruned.predict(X_test))
print(f"Training Accuracy (Post-pruned Tree): {post_prune_train_accuracy:.2f}")
print(f"Test Accuracy (Post-pruned Tree): {post_prune_test_accuracy:.2f}")

post_prune_recall = recall_score(y_test, dt_pruned.predict(X_test))
post_prune_auc = roc_auc_score(y_test, dt_pruned.predict_proba(X_test)[:, 1])
post_prune_conf_matrix = confusion_matrix(y_test, dt_pruned.predict(X_test))
fpr_post_prune, tpr_post_prune, _ = roc_curve(y_test,
dt_pruned.predict_proba(X_test)[:, 1])

plt.figure(figsize=(20, 10))
plot_tree(dt_pruned, feature_names=X.columns, class_names=['Not Survived',
'Survived'], filled=True)
plt.title("Decision Tree with Optimal Alpha for Titanic Dataset (Post-Pruning)")
plt.show()

#%%---------------Q7------------------
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score,
confusion_matrix,roc_curve

df = sns.load_dataset('titanic')
df.dropna(how='any', inplace=True)
X = df[['age', 'fare', 'parch', 'sibsp']]
y = df['survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5805, stratify=y)
lr_classifier = LogisticRegression(random_state=5805)
lr_classifier.fit(X_train, y_train)
logistic_regression_train_accuracy = accuracy_score(y_train,
lr_classifier.predict(X_train))
logistic_regression_test_accuracy = accuracy_score(y_test,
lr_classifier.predict(X_test))
print(f"Training Accuracy (Logistic Regression):
{logistic_regression_train_accuracy:.2f}")
print(f"Test Accuracy (Logistic Regression):
{logistic_regression_test_accuracy:.2f}")

logistic_regression_recall = recall_score(y_test, lr_classifier.predict(X_test))
logistic_regression_auc = roc_auc_score(y_test, lr_classifier.predict_proba(X_test)
[:, 1])
logistic_regression_conf_matrix = confusion_matrix(y_test,
lr_classifier.predict(X_test))

fpr_log_reg, tpr_log_reg, _ = roc_curve(y_test, lr_classifier.predict_proba(X_test)
[:, 1])
#%%---------------Q8------------------
from prettytable import PrettyTable, ALL
table = PrettyTable()
table.field_names = ["Index", "Model", "Train Accuracy", "Test Accuracy", "Recall",
"AUC", "Confusion Matrix"]
table.hrules = ALL
table.add_row([
    0, "Pre-Pruning", f"{pre_prune_train_accuracy:.2f}",
f"{pre_prune_test_accuracy:.2f}",
    f"{pre_prune_recall:.2f}", f"{pre_prune_auc:.2f}", f"{pre_prune_conf_matrix}"
])
table.add_row([
    1,"Post-Pruning", f"{post_prune_train_accuracy:.2f}",
f"{post_prune_test_accuracy:.2f}",
    f"{post_prune_recall:.2f}", f"{post_prune_auc:.2f}",
f"{post_prune_conf_matrix}"
])

table.add_row([
    2,"Logistic Regression", f"{logistic_regression_train_accuracy:.2f}",
f"{logistic_regression_test_accuracy:.2f}",
    f"{logistic_regression_recall:.2f}", f"{logistic_regression_auc:.2f}",
f"{logistic_regression_conf_matrix}"
])
print(table)

plt.figure(figsize=(10, 6))
plt.plot(fpr_pre_prune, tpr_pre_prune, label=f"DT Pre-Pruned (AUC =
```

```
{pre_prune_auc:.2f})")
plt.plot(fpr_post_prune, tpr_post_prune, label=f"DT Post-Pruned (AUC =
{post_prune_auc:.2f})")
plt.plot(fpr_log_reg, tpr_log_reg, label=f"Logistic Regression (AUC =
{logistic_regression_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```