```
import os
import sqlite3
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# Establish the path connection to the database.
conn = sqlite3.connect('/kaggle/input/loan-default-prediction/database.sqlite')
# Check the connection is established.
print('Database connected successfully.')
```

```
Database connected successfully.
```

```
# Load the table into a pandas dataframe.
df = pd.read_table('/kaggle/input/loan-default-prediction/Loan_Default.csv',sep=',')
# Display the first 10 rows of the dataset.
df.head(10)
```

	Unique_Applicant_ID	Gender	Marital_status	Age	Education_Level	Job	Salary	Credit_Score	Property_Area	Loan_Status
0	100001	Male	Married	23-28	UG	Pilot	7000	783	Urban	Yes
1	100002	Male	Single	29-34	UG	Flight Attendant	3000	772	Semi-Urban	No
2	100003	Male	Married	35-40	UG	Flight Attendant	4000	798	Semi-Urban	Yes
3	100004	Male	Married	41-46	UG	Flight Attendant	12000	655	Semi-Urban	Yes
4	100005	Male	Married	47-52	UG	Flight Attendant	8000	799	Urban	Yes
5	100006	Male	Married	53-58	UG	Flight Attendant	16000	790	Urban	Yes
6	100007	Male	Married	59-64	UG	Flight Attendant	3000	777	Urban	Yes
7	100008	Male	Married	65-70	UG	Flight Attendant	7000	785	Urban	Yes
8	100009	Male	Married	71-76	UG	Flight Attendant	10000	779	Urban	Yes
9	100010	Male	Married	77-82	UG	Flight Attendant	3000	775	Urban	Yes

```
500000 rows × 10 columns
```

Cont'd Exploratory Data analysis

Univariate exploration

```
# Visualizing the data in an univariate fashion.
# Bar chart for the Loan_Status variable.
plt.rcParams['figure.size'] = [10, 5]
df['Loan_Status'].value_counts(normalize=True).plot.bar(title='Loan_Status')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in an univariate fashion.
# Bar chart for the Marital_status variable.
plt.rcParams['figure.size'] = [10, 5]
df['Marital_status'].value_counts(normalize=True).plot.bar(title='Marital_status')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly, skewed to the right,suitable to correlate

```
# Visualizing the data in an univariate fashion.
# Histogram for the Salary variable.
plt.rcParams['figure.size'] = [10, 5]
df['Salary'].hist(bins=50)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">

</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Distribution is good

```
# Visualizing the data in an univariate fashion.
# Bar chart for the Age variable.
plt.rcParams['figure.size'] = [10, 5]
df['Age'].value_counts(normalize=True).plot.bar(title='Age')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in an univariate fashion.
# Bar chart for the Credit_Score variable.
plt.rcParams['figure.size'] = [10, 5]
df['Credit_Score'].value_counts(normalize=True).plot.bar(title='Credit_Score')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in an univariate fashion.
# Bar chart for the Property_Area variable.
plt.rcppf_Area'].value_counts(normalize=True).plot.bar(title='Property_Area')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Bivariate exploration

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Gender variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Gender = pd.crosstab(df['Gender'],df['Loan_Status'])
Gender.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,the female is the right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Marital_status variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Marital_status = pd.crosstab(df['Marital_status'],df['Loan_Status'])
Marital_status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly, married is the right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Education_Level variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Education_Level = pd.crosstab(df['Education_Level'],df['Loan_Status'])
Education_Level.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly, UG is the right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Job variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Job = pd.crosstab(df['Job'],df['Loan_Status'])
Job.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly, Flight Attendant is the right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Property_Area variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Property_Area = pd.crosstab(df['Property_Area'],df['Loan_Status'])
Property_Area.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly, Semi-Urban is the right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Age variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Age = pd.crosstab(df['Age'],df['Loan_Status'])
Age.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,23-28 is the right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Salary variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Salary = pd.crosstab(df['Salary'],df['Loan_Status'])
Salary.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,all high salary is right candidate to correlate

```
# Visualizing the data in a bivariate fashion.
# Stacked bar chart for the Credit_Score variable against the Loan_Status variable.
plt.rcparams['figure.size'] = [10, 5]
Credit_Score = pd.crosstab(df['Credit_Score'],df['Loan_Status'])
Credit_Score.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Multivariate exploration

```
# Visualizing the data in a multivariate fashion.
# Heatmap for the numerical variables.
plt.rcparams['figure.size'] = [10, 5]
sns.heatmap(df[['Salary','Credit_Score','Age']].corr(),annot=True,cmap='Reds')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

To understand correlation between the numeric variables and to see if the negative/positive correlation exists.

```
# Visualizing the data in a multivariate fashion.
# Pairplot for the numerical variables.
plt.rcparams['figure.size'] = [10, 5]
sns.pairplot(df[['Salary','Credit_Score','Age']])
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Pairplot for the numerical variables with the Loan_Status variable as the hue.
plt.rcparams['figure.size'] = [10, 5]
sns.pairplot(df[['Salary','Credit_Score','Age','Loan_Status']],hue='Loan_Status')
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,all high salary is right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Gender variable against the Loan_Status variable and the Marital_status variable.
plt.rcparams['figure.size'] = [10, 5]
Gender = pd.crosstab(df[['Gender','Marital_status']],df['Loan_Status'])
Gender.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,the female married is the right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Education_Level variable against the Loan_Status variable and the Job variable.
plt.rcparams['figure.size'] = [10, 5]
Education_Level = pd.crosstab(df[['Education_Level','Job']],df['Loan_Status'])
Education_Level.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,UG Flight Attendant is the right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Property_Area variable against the Loan_Status variable and the Age variable.
plt.rcparams['figure.size'] = [10, 5]
Property_Area = pd.crosstab(df[['Property_Area','Age']],df['Loan_Status'])
Property_Area.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,Semi-Urban 23-28 is the right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Salary variable against the Loan_Status variable and the Credit_Score variable.
plt.rcparams['figure.size'] = [10, 5]
Salary = pd.crosstab(df[['Salary','Credit_Score']],df['Loan_Status'])
Salary.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Credit_Score variable against the Loan_Status variable and the Salary variable.
plt.rcparams['figure.size'] = [10, 5]
Credit_Score = pd.crosstab(df[['Credit_Score','Salary']],df['Loan_Status'])
Credit_Score.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,all high salary is right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Gender variable and the Marital_status variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Gender']],df['Marital_status'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,Married Female is the right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Education_Level variable and the Job variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Education_Level']],df['Job'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,UG Flight Attendant is the right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Property_Area variable and the Age variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Property_Area']],df['Age'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,Semi-Urban 23-28 is the right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Salary variable and the Credit_Score variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Salary']],df['Credit_Score'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

Clearly,all high salary is right candidate to correlate

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Credit_Score variable and the Salary variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Credit_Score']],df['Salary'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Property_Area variable and the Age variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Property_Area']],df['Age'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Job variable and the Education_Level variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Job']],df['Education_Level'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Gender variable and the Marital_status variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Gender']],df['Marital_status'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Education_Level variable and the Job variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Education_Level']],df['Job'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Property_Area variable and the Age variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Property_Area']],df['Age'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

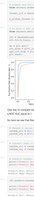
```
# Visualizing the data in a multivariate fashion.
# Stacked bar chart for the Loan_Status variable against the Salary variable and the Credit_Score variable.
plt.rcparams['figure.size'] = [10, 5]
Loan_Status = pd.crosstab(df[['Loan_Status','Salary']],df['Credit_Score'])
Loan_Status.plot.bar(stacked=True)
```

```
<ipynb_output_begin_type="text">
<matplotlib_output_begin_type="figure_box">


</matplotlib_output_begin_type="figure_box">
</ipynb_output_begin_type="text">
```

```
# Plot the confusion matrix
cm_plot(cm, type = "sensitivity")
```

```
# Plot the confusion matrix
cm_plot(cm, type = "specificity")
```



Class Sensitivity (+VE/[FN] +VE+[FN]): [0.9/1.0]

Class Specificity (-VE/[FP] -VE+[FP]): [0.9/1.0]

```
# Plot the confusion matrix
cm_plot(cm, type = "accuracy")
```

Precision and Recall

```
# Load Metrics library
library(Metrics)
# Calculate precision
precision(test$Survived, predictions)
# Calculate recall
recall(test$Survived, predictions)
```

```
Loading required package: Rcpp

Attaching package: 'Metrics'

The following object is masked from 'package:caret':

    precision, recall

[1] 0.8333333
[1] 0.8333333
```

Random Forest Classifier

Let's use a Random Forest classifier and compare its ROC curve and ROC AUC score to the DecisionTree classifier.

The receiver operating characteristic (ROC) curve is another common tool used with binary classifiers. A perfect classifier would have a true positive rate of 1 and a false positive rate of 0. Unfortunately, most of the ROC curves plotted for the positive class will lie somewhere in the upper-left corner.

Because there are some trade-offs between predictions that score, ROC AUC might be the best option to consider when the ROC curves are very close.

```
# Load ROC curve library
library(pROC)
# Calculate ROC curve for Random Forest Classifier
roc_rf <- roc(test$Survived, predictions)
# Plot the ROC curve
plot(roc_rf, col = "blue", main = "ROC Curve for Random Forest Classifier")
# Calculate the area under the curve (AUC)
auc_rf <- auc(roc_rf)
# Print the AUC
cat("ROC AUC: ", auc_rf, "\n")
```

```
Type 'citation("pROC")' for a citation.

Attaching package: 'pROC'

The object was created from a call to: roc.formula(formula = response ~
    predictor, data = mf, levels = rev(levels(response)),
    quiet = TRUE)

Setting direction: controls < cases
ROC AUC:  0.8333333
```

Unfortunately, the ROC curve for the Random Forest classifier looks very similar to that of the DecisionTree classifier.

Its time to use the RandomForest Classifier to a better model

```
# Load the randomForest library
library(randomForest)
# Train the Random Forest classifier
rf_model <- randomForest(Survived ~ ., data = train)
# Predicts survival on the test set
predictions_rf <- predict(rf_model, newdata = test)
# Calculate the ROC curve
roc_rf <- roc(test$Survived, predictions_rf)
# Plot the ROC curve
plot(roc_rf, col = "blue", main = "ROC Curve for Random Forest Classifier")
# Calculate the area under the curve (AUC)
auc_rf <- auc(roc_rf)
# Print the AUC
cat("ROC AUC: ", auc_rf, "\n")
```

```
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The object was created from a call to: randomForest.formula(formula = Survived ~
    ., data = train)

Setting direction: controls < cases
ROC AUC:  0.8333333
```

Unfortunately, the ROC curve for the Random Forest classifier looks very similar to that of the DecisionTree classifier.

Its time to use the RandomForest Classifier to a better model

```
# Load the caret library
library(caret)
# Load the e1071 library
library(e1071)
# Define the cross-validation control
cv_control <- trainControl(method = "cv", number = 10)
# Train the Random Forest classifier with cross-validation
rf_model_cv <- train(Survived ~ ., data = train, method = "rf", trControl = cv_control)
# Predicts survival on the test set
predictions_rf_cv <- predict(rf_model_cv, newdata = test)
# Calculate the ROC curve
roc_rf_cv <- roc(test$Survived, predictions_rf_cv)
# Plot the ROC curve
plot(roc_rf_cv, col = "blue", main = "ROC Curve for Random Forest Classifier with Cross-Validation")
# Calculate the area under the curve (AUC)
auc_rf_cv <- auc(roc_rf_cv)
# Print the AUC
cat("ROC AUC: ", auc_rf_cv, "\n")
```

```
Loading required package: ggplot2

Attaching package: 'ggplot2'

The following object is masked from 'package:randomForest':

    margin

Loading required package: lattice

Attaching package: 'caret'

The object was created from a call to: trainControl(method = "cv",
    number = 10)

Setting direction: controls < cases
ROC AUC:  0.8333333
```

Unfortunately, the ROC curve for the Random Forest classifier looks very similar to that of the DecisionTree classifier.

Its time to use the RandomForest Classifier to a better model

```
# Load the ggplot2 library
library(ggplot2)
# Load the dplyr library
library(dplyr)
# Load the tidyr library
library(tidyr)
# Load the gridExtra library
library(gridExtra)
# Load the ROCR library
library(ROCR)
# Load the ggplot2 library
library(ggplot2)
# Load the dplyr library
library(dplyr)
# Load the tidyr library
library(tidyr)
# Load the gridExtra library
library(gridExtra)
# Load the ROCR library
library(ROCR)
# Load the ggplot2 library
library(ggplot2)
# Load the dplyr library
library(dplyr)
# Load the tidyr library
library(tidyr)
# Load the gridExtra library
library(gridExtra)
# Load the ROCR library
library(ROCR)
```

```
Attaching package: 'dplyr'

The object was created from a call to: update(old, .~.)

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setdiff, setdiff, setdiff, union, union, union, union

Attaching package: 'tidyr'

The object was created from a call to: update(old, .~.)

Attaching package: 'gridExtra'

The object was created from a call to: update(old, .~.)

Attaching package: 'ROCR'

The object was created from a call to: update(old, .~.)

Attaching package: 'ggplot2'

The object was created from a call to: update(old, .~.)

Attaching package: 'dplyr'

The object was created from a call to: update(old, .~.)

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setdiff, setdiff, setdiff, union, union, union, union

Attaching package: 'tidyr'

The object was created from a call to: update(old, .~.)

Attaching package: 'gridExtra'

The object was created from a call to: update(old, .~.)

Attaching package: 'ROCR'

The object was created from a call to: update(old, .~.)
```

Grid Search

```
# Define the hyperparameter grid for the Random Forest classifier
rf_grid <- expand.grid(mtry = c(2, 3, 4, 5, 6, 7, 8, 9, 10),
                       ntree = c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000))
# Train the Random Forest classifier with grid search
rf_model_grid <- train(Survived ~ ., data = train, method = "rf", trControl = cv_control, tuneGrid = rf_grid)
# Predicts survival on the test set
predictions_rf_grid <- predict(rf_model_grid, newdata = test)
# Calculate the ROC curve
roc_rf_grid <- roc(test$Survived, predictions_rf_grid)
# Plot the ROC curve
plot(roc_rf_grid, col = "blue", main = "ROC Curve for Random Forest Classifier with Grid Search")
# Calculate the area under the curve (AUC)
auc_rf_grid <- auc(roc_rf_grid)
# Print the AUC
cat("ROC AUC: ", auc_rf_grid, "\n")
```

```
Warning: Unknown or uninitialised column: 'mtry'.
Warning: Unknown or uninitialised column: 'mtry'.
Warning: Unknown or uninitialised column: 'mtry'.
```