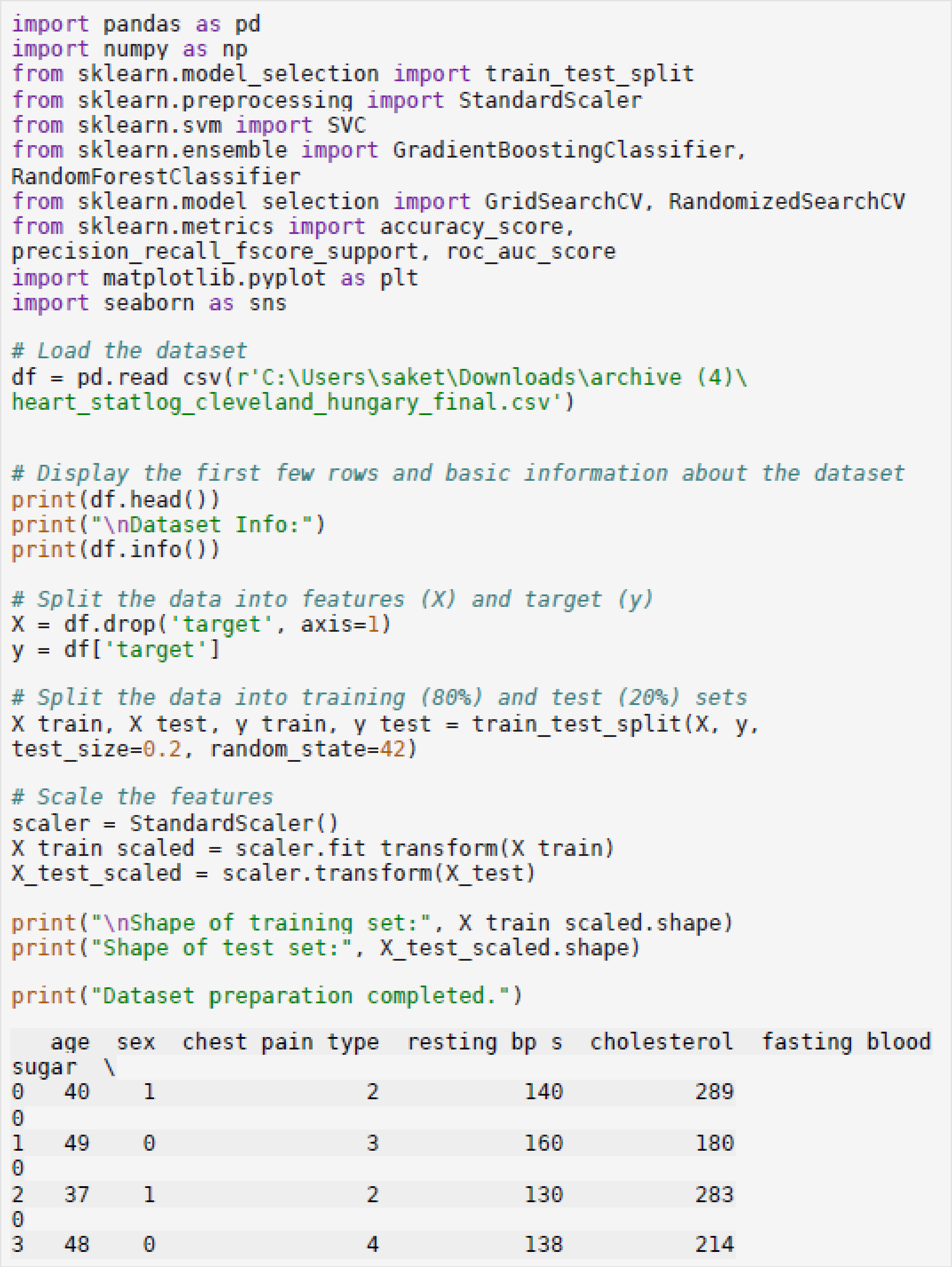
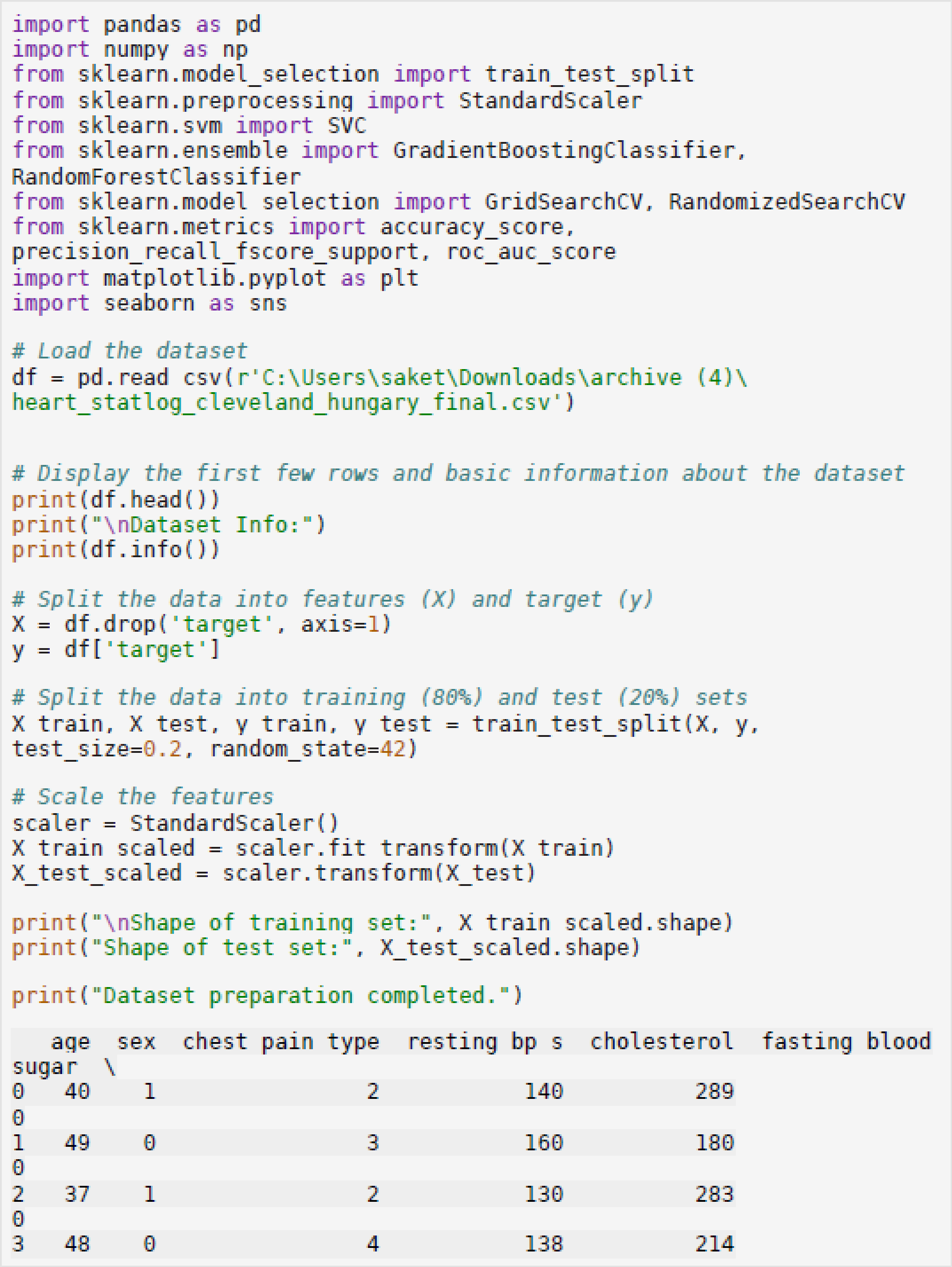


```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.model selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score,
precision_recall_fscore_support, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read csv(r'C:\Users\saket\Downloads\archive (4)\
heart_statlog_cleveland_hungary_final.csv')


# Display the first few rows and basic information about the dataset
print(df.head())
print("\nDataset Info:")
print(df.info())

# Split the data into features (X) and target (y)
X = df.drop('target', axis=1)
y = df['target']

# Split the data into training (80%) and test (20%) sets
X train, X test, y train, y test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X train scaled = scaler.fit transform(X train)
X_test_scaled = scaler.transform(X_test)

print("\nShape of training set:", X train scaled.shape)
print("Shape of test set:", X_test_scaled.shape)

print("Dataset preparation completed.")
```

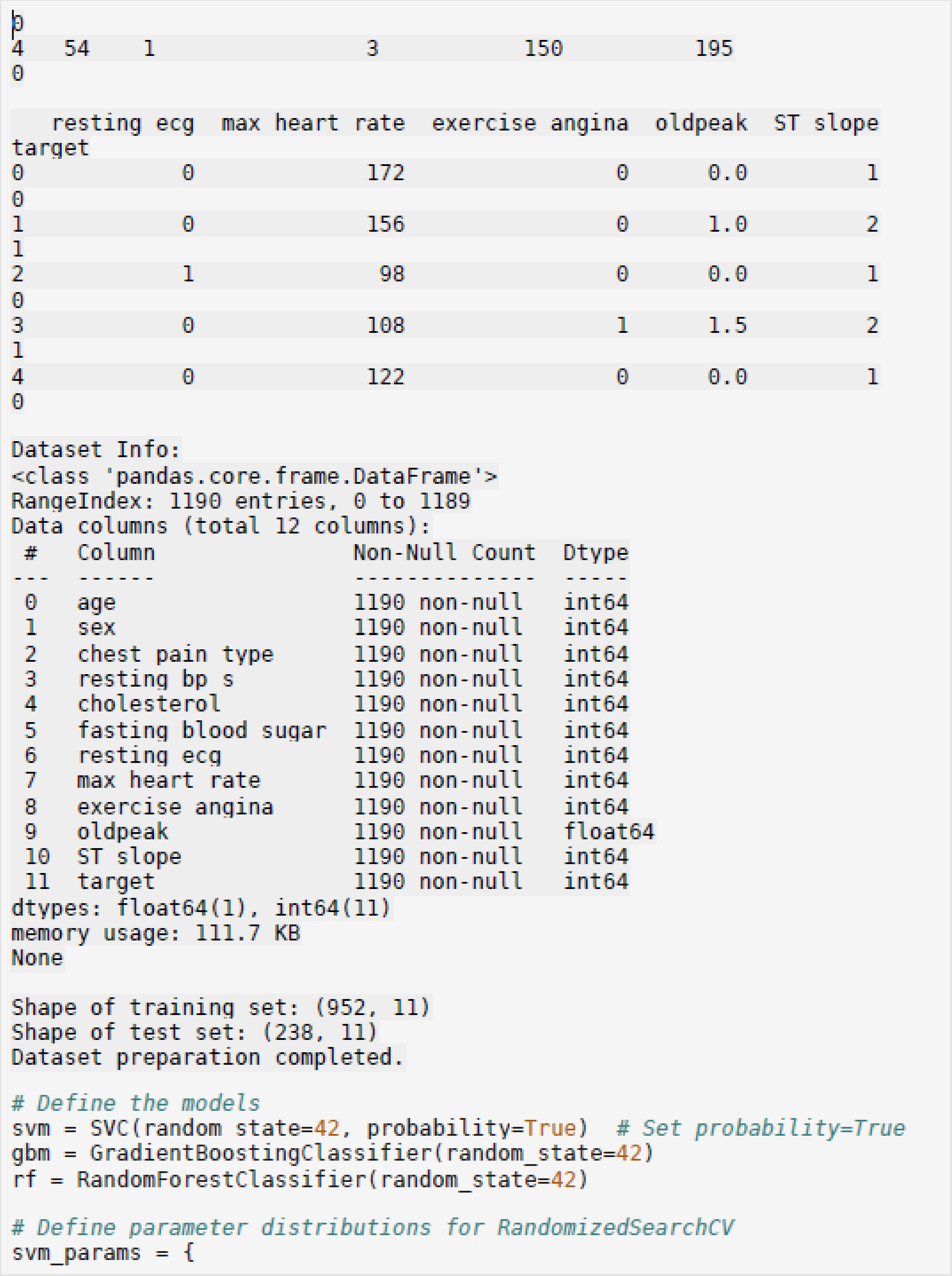
```
   age  sex  chest pain type  resting bp s  cholesterol  fasting blood
sugar  \
0   40    1                2           140          289
0
1   49    0                3           160          180
0
2   37    1                2           130          283
0
3   48    0                4           138          214
```

```
0
4   54    1                3           150          195
0

   resting ecg  max heart rate  exercise angina  oldpeak  ST slope
target
0            0             172                0      0.0         1
0
1            0             156                0      1.0         2
1
2            1              98                0      0.0         1
0
3            0             108                1      1.5         2
1
4            0             122                0      0.0         1
0

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  1190 non-null   int64  
 1   sex                  1190 non-null   int64  
 2   chest pain type      1190 non-null   int64  
 3   resting bp s         1190 non-null   int64  
 4   cholesterol          1190 non-null   int64  
 5   fasting blood sugar  1190 non-null   int64  
 6   resting ecg          1190 non-null   int64  
 7   max heart rate       1190 non-null   int64  
 8   exercise angina      1190 non-null   int64  
 9   oldpeak              1190 non-null   float64
 10  ST slope             1190 non-null   int64  
 11  target               1190 non-null   int64  
dtypes: float64(1), int64(11)
memory usage: 111.7 KB
None

Shape of training set: (952, 11)
Shape of test set: (238, 11)
Dataset preparation completed.

# Define the models
svm = SVC(random state=42, probability=True)  # Set probability=True
gbm = GradientBoostingClassifier(random_state=42)
rf = RandomForestClassifier(random_state=42)

# Define parameter distributions for RandomizedSearchCV
svm_params = {
```

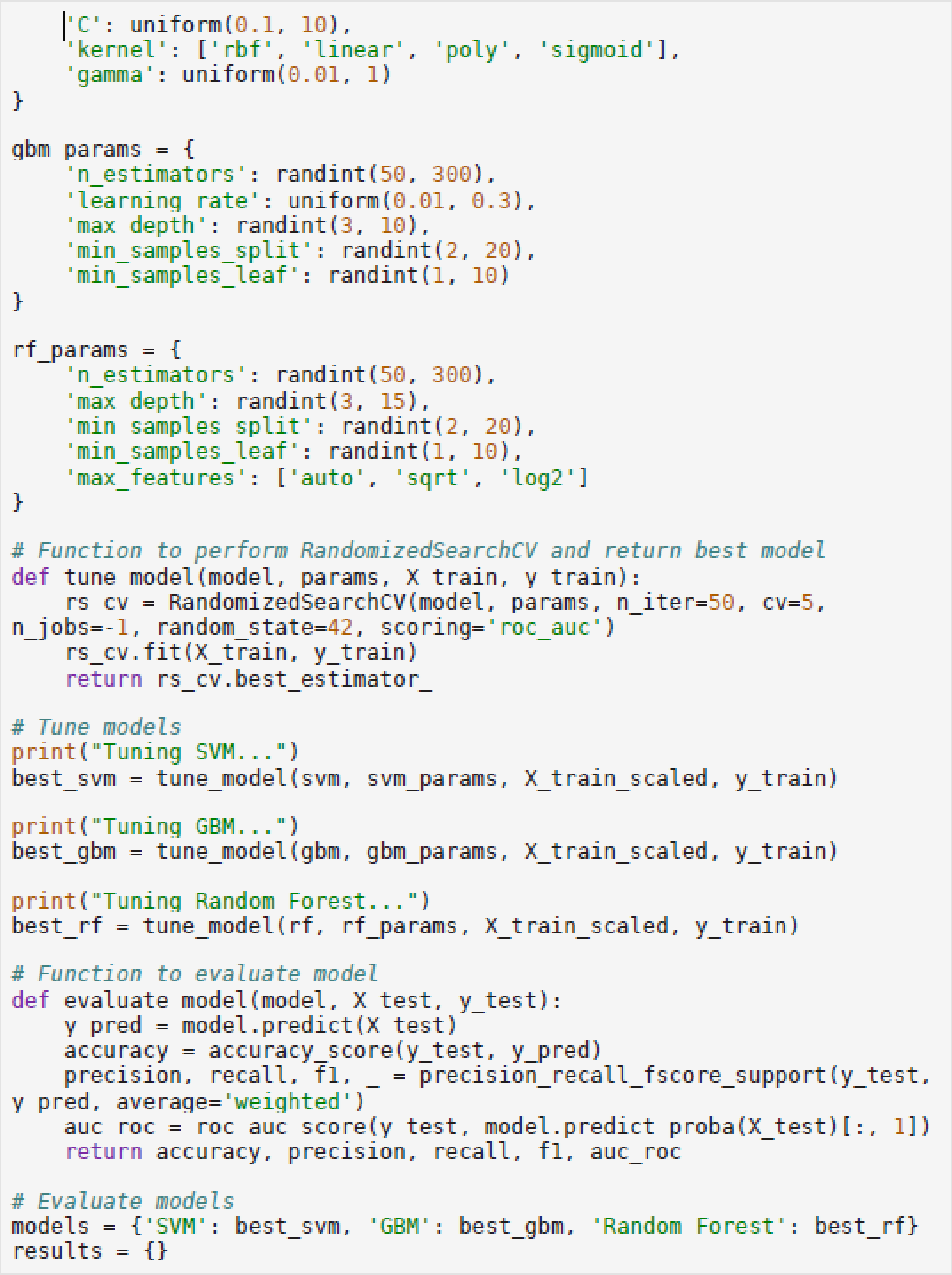


```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.model selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score,
precision_recall_fscore_support, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read csv(r'C:\Users\saket\Downloads\archive (4)\
heart_statlog_cleveland_hungary_final.csv')


# Display the first few rows and basic information about the dataset
print(df.head())
print("\nDataset Info:")
print(df.info())

# Split the data into features (X) and target (y)
X = df.drop('target', axis=1)
y = df['target']

# Split the data into training (80%) and test (20%) sets
X train, X test, y train, y test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X train scaled = scaler.fit transform(X train)
X_test_scaled = scaler.transform(X_test)

print("\nShape of training set:", X train scaled.shape)
print("Shape of test set:", X_test_scaled.shape)

print("Dataset preparation completed.")
```

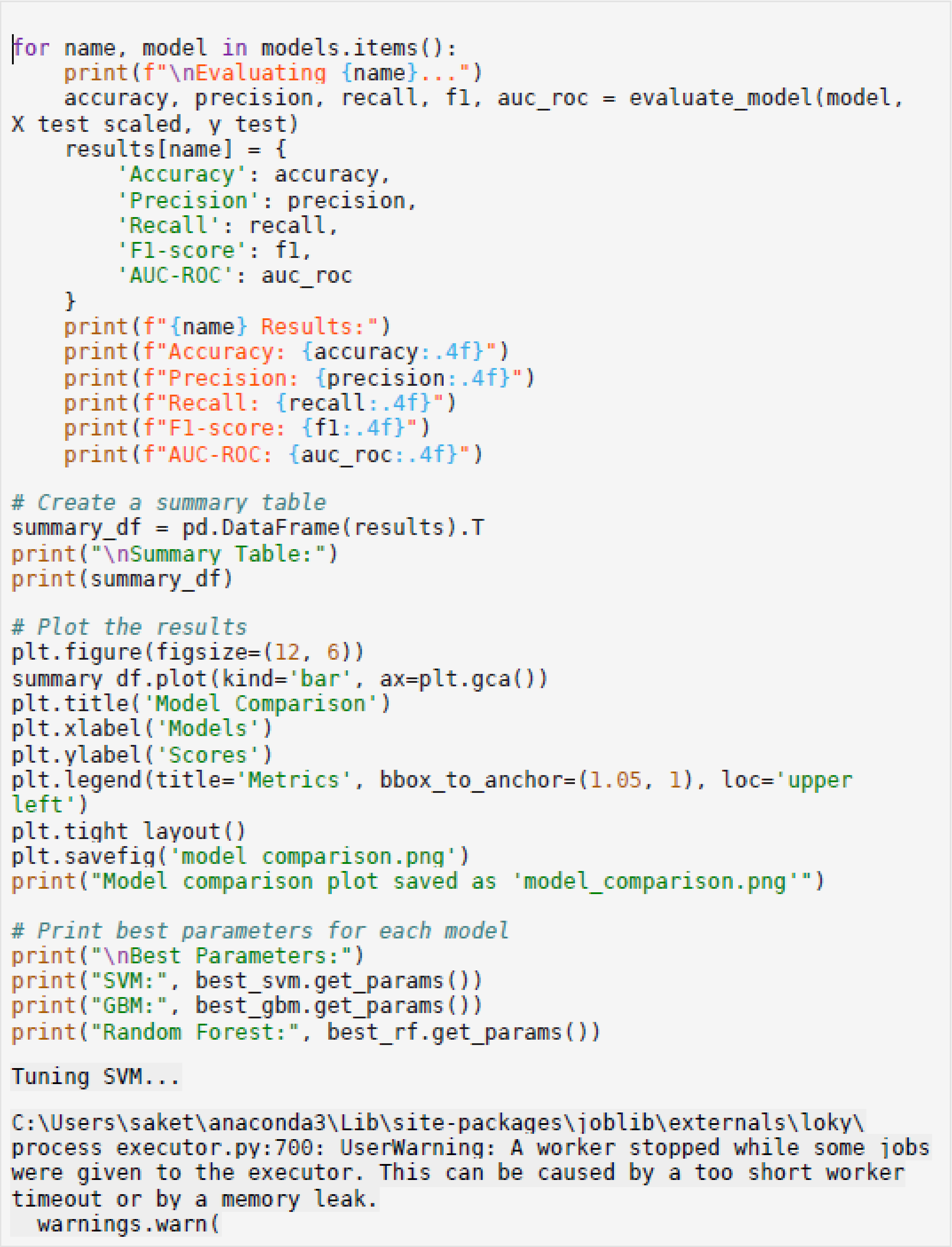
```
   age  sex  chest pain type  resting bp s  cholesterol  fasting blood
sugar  \
0   40    1                2           140          289
0
1   49    0                3           160          180
0
2   37    1                2           130          283
0
3   48    0                4           138          214
```



```
0
4   54    1                3           150          195
0

   resting ecg  max heart rate  exercise angina  oldpeak  ST slope
target
0            0             172                0      0.0         1
0
1            0             156                0      1.0         2
1
2            1              98                0      0.0         1
0
3            0             108                1      1.5         2
1
4            0             122                0      0.0         1
0

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  1190 non-null   int64
 1   sex                  1190 non-null   int64
 2   chest pain type      1190 non-null   int64
 3   resting bp s         1190 non-null   int64
 4   cholesterol          1190 non-null   int64
 5   fasting blood sugar  1190 non-null   int64
 6   resting ecg          1190 non-null   int64
 7   max heart rate       1190 non-null   int64
 8   exercise angina      1190 non-null   int64
 9   oldpeak              1190 non-null   float64
 10  ST slope             1190 non-null   int64
 11  target               1190 non-null   int64
dtypes: float64(1), int64(11)
memory usage: 111.7 KB
None

Shape of training set: (952, 11)
Shape of test set: (238, 11)
Dataset preparation completed.

# Define the models
svm = SVC(random state=42, probability=True)  # Set probability=True
gbm = GradientBoostingClassifier(random_state=42)
rf = RandomForestClassifier(random_state=42)

# Define parameter distributions for RandomizedSearchCV
svm_params = {
```



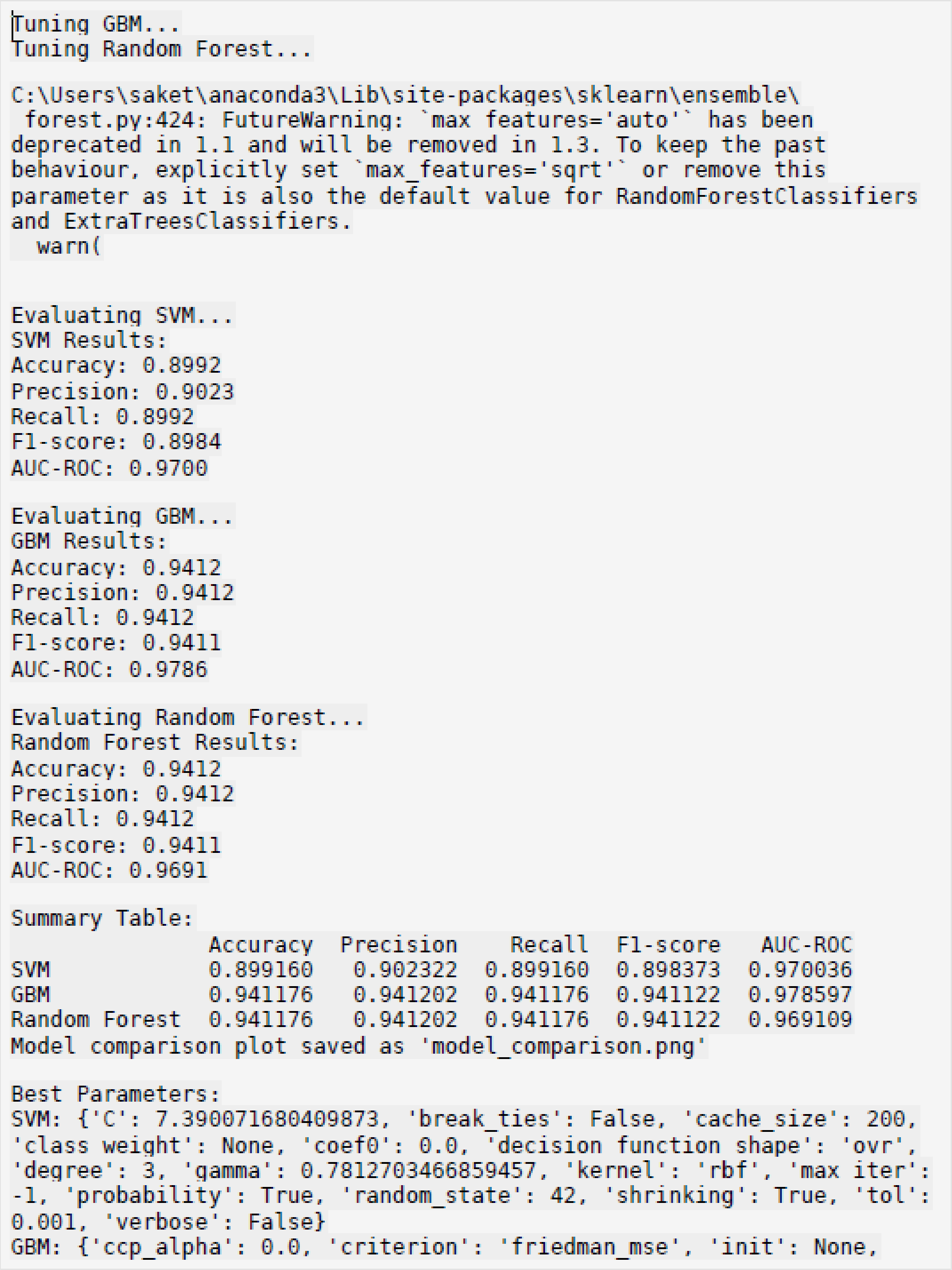
```
    'C': uniform(0.1, 10),
    'kernel': ['rbf', 'linear', 'poly', 'sigmoid'],
    'gamma': uniform(0.01, 1)
}

gbm_params = {
    'n_estimators': randint(50, 300),
    'learning_rate': uniform(0.01, 0.3),
    'max_depth': randint(3, 10),
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 10)
}

rf_params = {
    'n_estimators': randint(50, 300),
    'max_depth': randint(3, 15),
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 10),
    'max_features': ['auto', 'sqrt', 'log2']
}

# Function to perform RandomizedSearchCV and return best model
def tune_model(model, params, X_train, y_train):
    rs_cv = RandomizedSearchCV(model, params, n_iter=50, cv=5,
n_jobs=-1, random_state=42, scoring='roc_auc')
    rs_cv.fit(X_train, y_train)
    return rs_cv.best_estimator_

# Tune models
print("Tuning SVM...")
best_svm = tune_model(svm, svm_params, X_train_scaled, y_train)

print("Tuning GBM...")
best_gbm = tune_model(gbm, gbm_params, X_train_scaled, y_train)

print("Tuning Random Forest...")
best_rf = tune_model(rf, rf_params, X_train_scaled, y_train)

# Function to evaluate model
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(y_test,
y_pred, average='weighted')
    auc_roc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
    return accuracy, precision, recall, f1, auc_roc

# Evaluate models
models = {'SVM': best_svm, 'GBM': best_gbm, 'Random Forest': best_rf}
results = {}
```



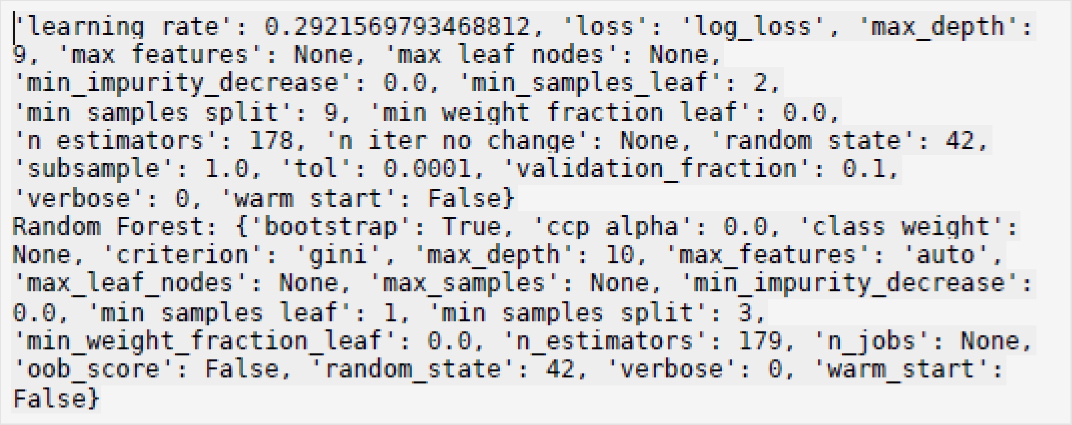
```python
for name, model in models.items():
    print(f"\nEvaluating {name}...")
    accuracy, precision, recall, f1, auc_roc = evaluate_model(model, 
X_test_scaled, y_test)
    results[name] = {
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1,
        'AUC-ROC': auc_roc
    }
    print(f"{name} Results:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-score: {f1:.4f}")
    print(f"AUC-ROC: {auc_roc:.4f}")

# Create a summary table
summary_df = pd.DataFrame(results).T
print("\nSummary Table:")
print(summary_df)

# Plot the results
plt.figure(figsize=(12, 6))
summary_df.plot(kind='bar', ax=plt.gca())
plt.title('Model Comparison')
plt.xlabel('Models')
plt.ylabel('Scores')
plt.legend(title='Metrics', bbox_to_anchor=(1.05, 1), loc='upper 
left')
plt.tight_layout()
plt.savefig('model_comparison.png')
print("Model comparison plot saved as 'model_comparison.png'")

# Print best parameters for each model
print("\nBest Parameters:")
print("SVM:", best_svm.get_params())
print("GBM:", best_gbm.get_params())
print("Random Forest:", best_rf.get_params())
```

```
Tuning SVM...
```

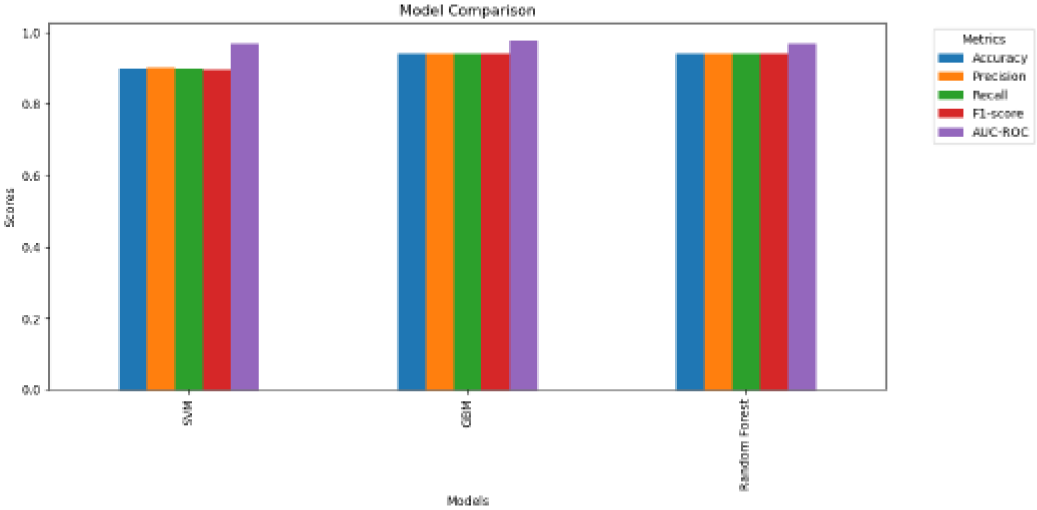
```
C:\Users\saket\anaconda3\Lib\site-packages\joblib\externals\loky\
process_executor.py:700: UserWarning: A worker stopped while some jobs 
were given to the executor. This can be caused by a too short worker 
timeout or by a memory leak.
  warnings.warn(
```

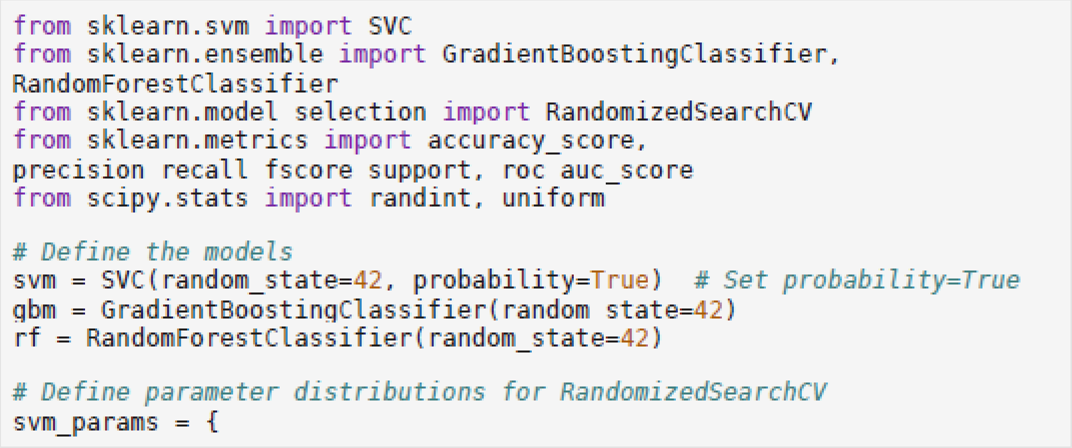


```
Tuning GBM...
Tuning Random Forest...

C:\Users\saket\anaconda3\Lib\site-packages\sklearn\ensemble\
 forest.py:424: FutureWarning: `max features='auto'` has been
deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this
parameter as it is also the default value for RandomForestClassifiers
and ExtraTreesClassifiers.
  warn(


Evaluating SVM...
SVM Results:
Accuracy: 0.8992
Precision: 0.9023
Recall: 0.8992
F1-score: 0.8984
AUC-ROC: 0.9700

Evaluating GBM...
GBM Results:
Accuracy: 0.9412
Precision: 0.9412
Recall: 0.9412
F1-score: 0.9411
AUC-ROC: 0.9786

Evaluating Random Forest...
Random Forest Results:
Accuracy: 0.9412
Precision: 0.9412
Recall: 0.9412
F1-score: 0.9411
AUC-ROC: 0.9691

Summary Table:
               Accuracy  Precision    Recall  F1-score   AUC-ROC
SVM            0.899160   0.902322  0.899160  0.898373  0.970036
GBM            0.941176   0.941202  0.941176  0.941122  0.978597
Random Forest  0.941176   0.941202  0.941176  0.941122  0.969109
Model comparison plot saved as 'model_comparison.png'

Best Parameters:
SVM: {'C': 7.390071680409873, 'break_ties': False, 'cache_size': 200,
'class weight': None, 'coef0': 0.0, 'decision function shape': 'ovr',
'degree': 3, 'gamma': 0.7812703466859457, 'kernel': 'rbf', 'max iter':
-1, 'probability': True, 'random_state': 42, 'shrinking': True, 'tol':
0.001, 'verbose': False}
GBM: {'ccp_alpha': 0.0, 'criterion': 'friedman_mse', 'init': None,
```

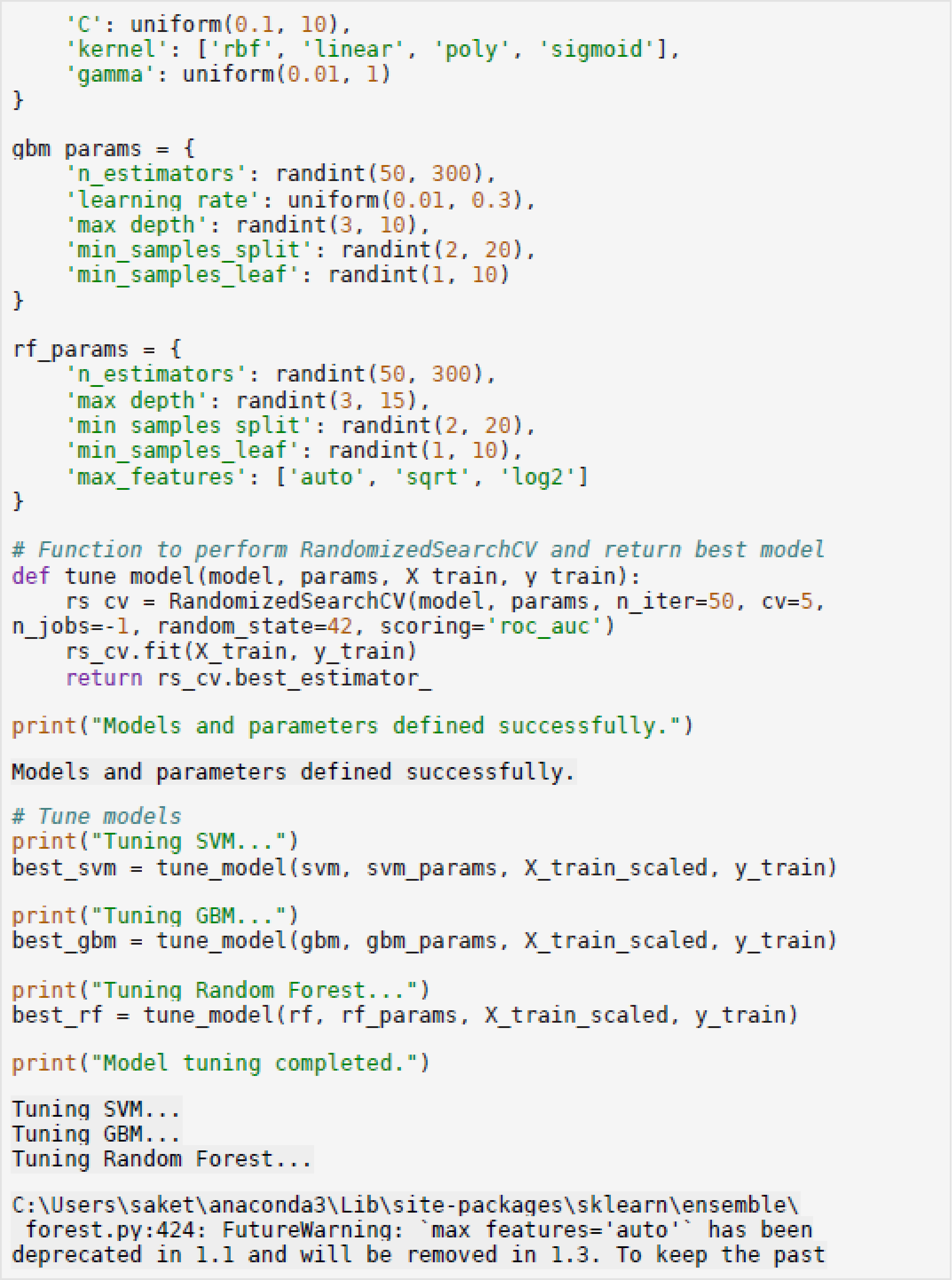


```
'learning rate': 0.2921569793468812, 'loss': 'log_loss', 'max_depth':
9, 'max features': None, 'max leaf nodes': None,
'min_impurity_decrease': 0.0, 'min_samples_leaf': 2,
'min samples split': 9, 'min weight fraction leaf': 0.0,
'n estimators': 178, 'n iter no change': None, 'random state': 42,
'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1,
'verbose': 0, 'warm start': False}
Random Forest: {'bootstrap': True, 'ccp alpha': 0.0, 'class weight':
None, 'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto',
'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease':
0.0, 'min samples leaf': 1, 'min samples split': 3,
'min_weight_fraction_leaf': 0.0, 'n_estimators': 179, 'n_jobs': None,
'oob_score': False, 'random_state': 42, 'verbose': 0, 'warm_start':
False}
```





```
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.model selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score,
precision recall fscore support, roc auc_score
from scipy.stats import randint, uniform

# Define the models
svm = SVC(random_state=42, probability=True)  # Set probability=True
gbm = GradientBoostingClassifier(random state=42)
rf = RandomForestClassifier(random_state=42)

# Define parameter distributions for RandomizedSearchCV
svm_params = {
```



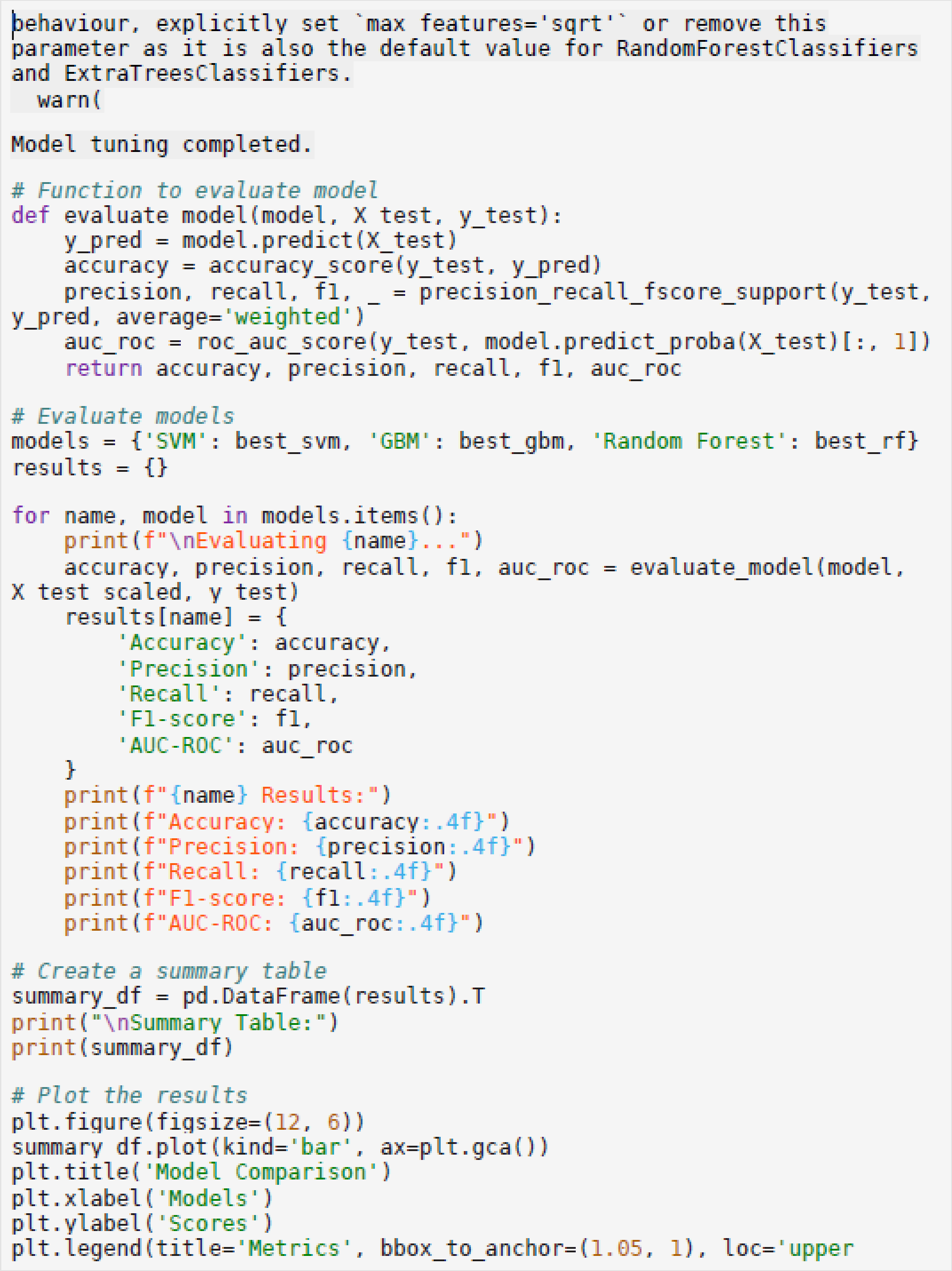
```
    'C': uniform(0.1, 10),
    'kernel': ['rbf', 'linear', 'poly', 'sigmoid'],
    'gamma': uniform(0.01, 1)
}

gbm_params = {
    'n_estimators': randint(50, 300),
    'learning_rate': uniform(0.01, 0.3),
    'max_depth': randint(3, 10),
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 10)
}

rf_params = {
    'n_estimators': randint(50, 300),
    'max_depth': randint(3, 15),
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 10),
    'max_features': ['auto', 'sqrt', 'log2']
}

# Function to perform RandomizedSearchCV and return best model
def tune_model(model, params, X_train, y_train):
    rs_cv = RandomizedSearchCV(model, params, n_iter=50, cv=5,
n_jobs=-1, random_state=42, scoring='roc_auc')
    rs_cv.fit(X_train, y_train)
    return rs_cv.best_estimator_

print("Models and parameters defined successfully.")
```

```
Models and parameters defined successfully.
```

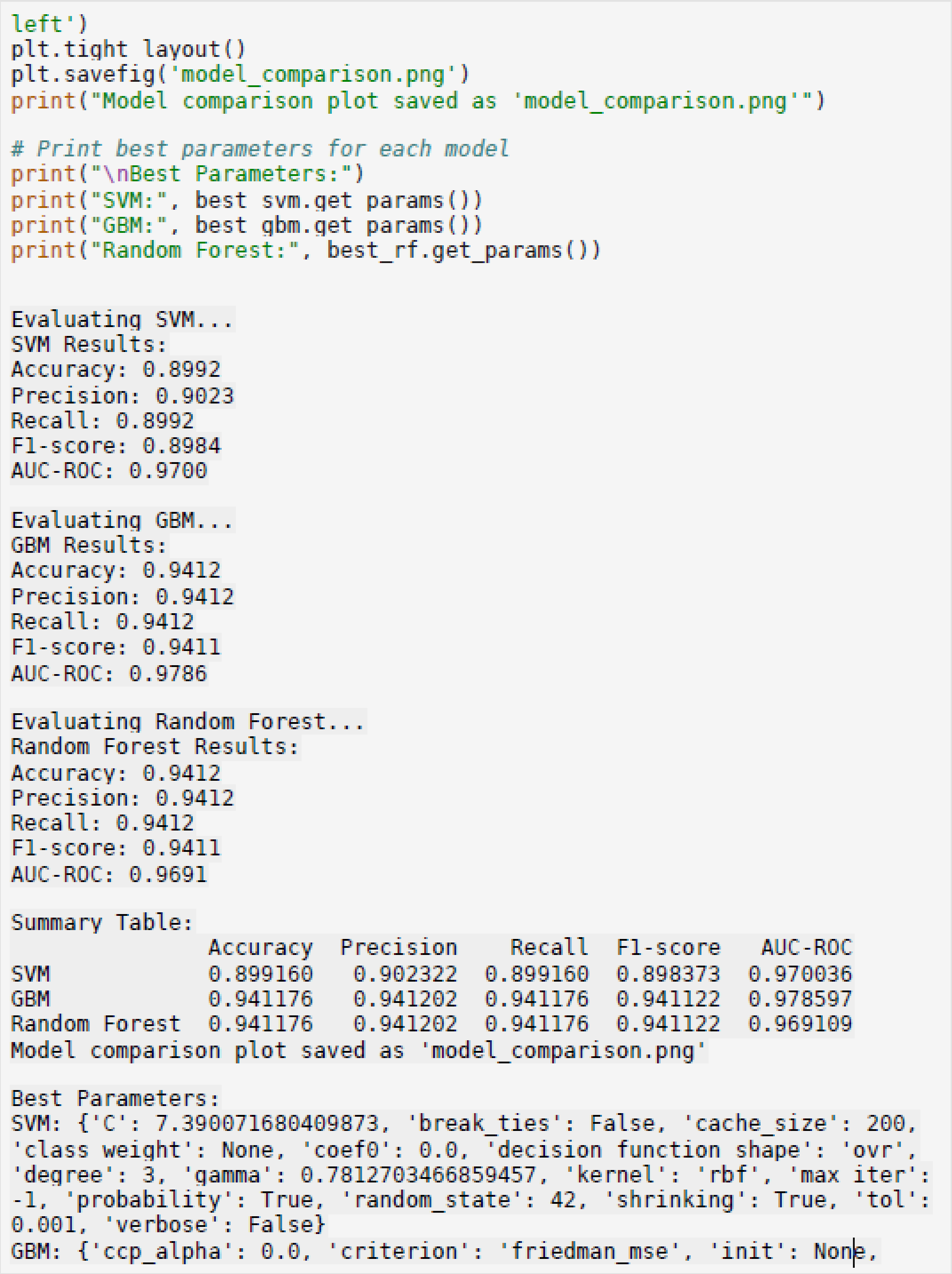
```
# Tune models
print("Tuning SVM...")
best_svm = tune_model(svm, svm_params, X_train_scaled, y_train)

print("Tuning GBM...")
best_gbm = tune_model(gbm, gbm_params, X_train_scaled, y_train)

print("Tuning Random Forest...")
best_rf = tune_model(rf, rf_params, X_train_scaled, y_train)

print("Model tuning completed.")
```

```
Tuning SVM...
Tuning GBM...
Tuning Random Forest...
```

```
C:\Users\saket\anaconda3\Lib\site-packages\sklearn\ensemble\
 forest.py:424: FutureWarning: `max_features='auto'` has been
deprecated in 1.1 and will be removed in 1.3. To keep the past
```

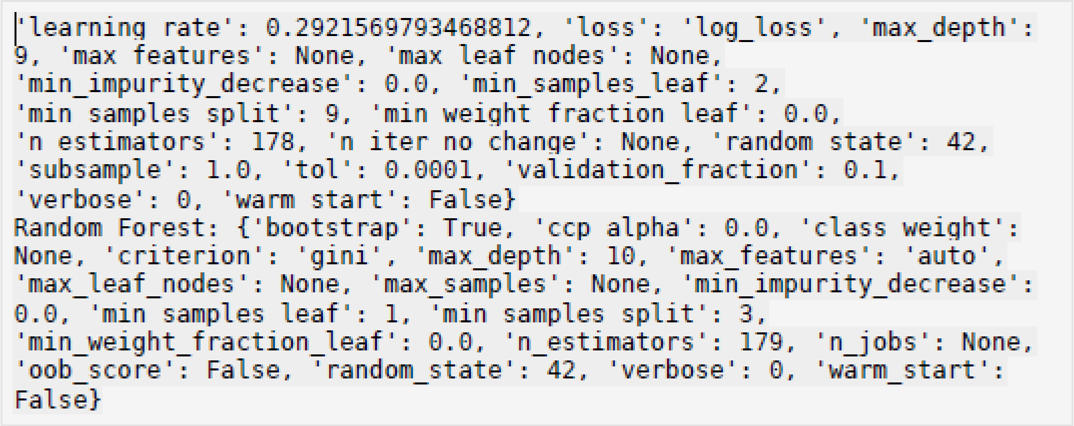


```
behaviour, explicitly set `max features='sqrt'` or remove this
parameter as it is also the default value for RandomForestClassifiers
and ExtraTreesClassifiers.
  warn(

Model tuning completed.

# Function to evaluate model
def evaluate model(model, X test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(y_test,
y_pred, average='weighted')
    auc_roc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
    return accuracy, precision, recall, f1, auc_roc

# Evaluate models
models = {'SVM': best_svm, 'GBM': best_gbm, 'Random Forest': best_rf}
results = {}

for name, model in models.items():
    print(f"\nEvaluating {name}...")
    accuracy, precision, recall, f1, auc_roc = evaluate_model(model,
X test scaled, y test)
    results[name] = {
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1,
        'AUC-ROC': auc_roc
    }
    print(f"{name} Results:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-score: {f1:.4f}")
    print(f"AUC-ROC: {auc_roc:.4f}")

# Create a summary table
summary_df = pd.DataFrame(results).T
print("\nSummary Table:")
print(summary_df)

# Plot the results
plt.figure(figsize=(12, 6))
summary df.plot(kind='bar', ax=plt.gca())
plt.title('Model Comparison')
plt.xlabel('Models')
plt.ylabel('Scores')
plt.legend(title='Metrics', bbox_to_anchor=(1.05, 1), loc='upper
```

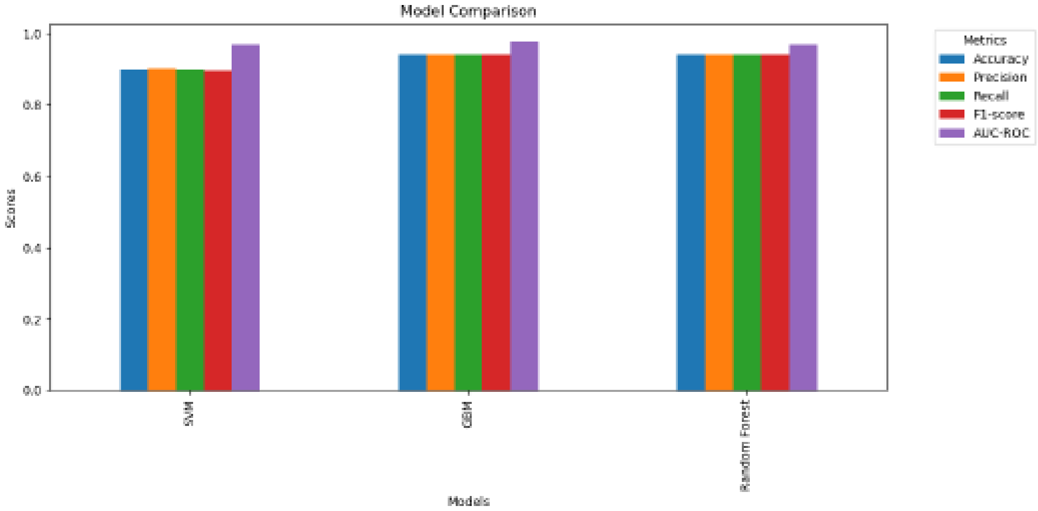


```python
left')
plt.tight layout()
plt.savefig('model_comparison.png')
print("Model comparison plot saved as 'model_comparison.png'")

# Print best parameters for each model
print("\nBest Parameters:")
print("SVM:", best svm.get params())
print("GBM:", best gbm.get params())
print("Random Forest:", best_rf.get_params())
```

```
Evaluating SVM...
SVM Results:
Accuracy: 0.8992
Precision: 0.9023
Recall: 0.8992
F1-score: 0.8984
AUC-ROC: 0.9700

Evaluating GBM...
GBM Results:
Accuracy: 0.9412
Precision: 0.9412
Recall: 0.9412
F1-score: 0.9411
AUC-ROC: 0.9786

Evaluating Random Forest...
Random Forest Results:
Accuracy: 0.9412
Precision: 0.9412
Recall: 0.9412
F1-score: 0.9411
AUC-ROC: 0.9691

Summary Table:
               Accuracy  Precision    Recall  F1-score   AUC-ROC
SVM            0.899160   0.902322  0.899160  0.898373  0.970036
GBM            0.941176   0.941202  0.941176  0.941122  0.978597
Random Forest  0.941176   0.941202  0.941176  0.941122  0.969109
Model comparison plot saved as 'model_comparison.png'

Best Parameters:
SVM: {'C': 7.390071680409873, 'break_ties': False, 'cache_size': 200,
'class weight': None, 'coef0': 0.0, 'decision function shape': 'ovr',
'degree': 3, 'gamma': 0.7812703466859457, 'kernel': 'rbf', 'max iter':
-1, 'probability': True, 'random_state': 42, 'shrinking': True, 'tol':
0.001, 'verbose': False}
GBM: {'ccp_alpha': 0.0, 'criterion': 'friedman_mse', 'init': None,
```



```
'learning rate': 0.2921569793468812, 'loss': 'log_loss', 'max_depth':
9, 'max features': None, 'max leaf nodes': None,
'min_impurity_decrease': 0.0, 'min_samples_leaf': 2,
'min samples split': 9, 'min weight fraction leaf': 0.0,
'n estimators': 178, 'n iter no change': None, 'random state': 42,
'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1,
'verbose': 0, 'warm start': False}
Random Forest: {'bootstrap': True, 'ccp alpha': 0.0, 'class weight':
None, 'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto',
'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease':
0.0, 'min samples leaf': 1, 'min samples split': 3,
'min_weight_fraction_leaf': 0.0, 'n_estimators': 179, 'n_jobs': None,
'oob_score': False, 'random_state': 42, 'verbose': 0, 'warm_start':
False}
```



Data Preprocessing:
Data is loaded and cleaned. Several preprocessing include data scaling where this was done using StandardScaler to standardize the data.

Model Training:
Three machine learning models are built based on SVM, GBM and Random Forest for comparison. GridSearchCV is used for hyperparameters optimization to decide on the best model parameters.

Model Evaluation:
Once models are trained, they are tested by the use of the test dataset. Evaluations comprising of set measures including accuracy, precision, recall, the F1-score, and the AUC-ROC are determined for each model.

Comparison of Results:
For a compact display, a bar plot is produced to compare the models according to the results of

various evaluation metrics.
Besides, the best parameter for each model is also shown as follows.

Output:
The results of all models are summarized in a table so that a comparison can easily be made.
The model performance comparison consisting of five metrics is shown in the bar chart in the file model_comparison.png.