```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
import pandas as pd

# Step 1: Data Preparation

# Load your dataset
data = pd.read_csv('emails.csv')

print(data.columns)
print(data.head())
```

```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a',
'you', 'hou',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure',
'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
  Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey
jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  ...         0
0
1   Email 2    8  13   24    6    6   2  102    1   27  ...         0
0
2   Email 3    0   0    1    0    0   0    8    0    0  ...         0
0
3   Email 4    0   5   22    0    5   1   51    2   10  ...         0
0
4   Email 5    7   6   17    1    5   2   57    0    9  ...         0
0

   valued  lay  infrastructure  military  allowing  ff  dry
Prediction
0       0    0               0         0         0   0    0
0
1       0    0               0         0         0   1    0
0
2       0    0               0         0         0   0    0
0
3       0    0               0         0         0   0    0
```

```
0
4       0    0                0         0         0   1    0
0

[5 rows x 3002 columns]
```

```
# Assign labels to y (target variable)
y = data['Prediction']

# Prepare features (X)
X = data.drop(columns=['Email No.', 'Prediction'])

# Handle missing values
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

# Step 2: Model Selection

# Initialize classifiers
classifiers = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Support Vector Machine': SVC(),
    'Random Forest': RandomForestClassifier(),
    'Gradient Boosting': GradientBoostingClassifier()
}

# Step 3: Model Training and Evaluation

for name, clf in classifiers.items():
    print(f"Training and evaluating {name}...")

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

    # Train the classifier
    clf.fit(X_train, y_train)
```

```
Training and evaluating Logistic Regression...

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
```

```
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

Training and evaluating Decision Tree...
Training and evaluating Support Vector Machine...
Training and evaluating Random Forest...
Training and evaluating Gradient Boosting...
```

```python
# Model Training and Evaluation
# Model Training and Evaluation
for name, clf in classifiers.items():
    print(f"Training and evaluating {name}...")
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)

    # Evaluation Metrics
    accuracy = accuracy_score(y_test, predictions)
    precision = precision_score(y_test, predictions)
    recall = recall_score(y_test, predictions, pos_label=1)  #
Assuming positive label is 1
    f1 = f1_score(y_test, predictions, pos_label=1)  # Assuming
positive label is 1

    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-Score: {f1:.2f}")
    print()
```

```
Training and evaluating Logistic Regression...

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
Accuracy: 0.97
Precision: 0.93
Recall: 0.97
F1-Score: 0.95

Training and evaluating Decision Tree...
Accuracy: 0.93
Precision: 0.88
Recall: 0.87
F1-Score: 0.87

Training and evaluating Support Vector Machine...
Accuracy: 0.95
Precision: 1.00
Recall: 0.82
F1-Score: 0.90

Training and evaluating Random Forest...
Accuracy: 0.98
Precision: 0.95
Recall: 0.97
F1-Score: 0.96

Training and evaluating Gradient Boosting...
Accuracy: 0.97
Precision: 0.94
Recall: 0.96
F1-Score: 0.95


# Step 4: Model Evaluation with Cross-Validation

for name, clf in classifiers.items():
    print(f"Evaluating {name} with cross-validation...")

    # Use cross-validation to evaluate the model's generalization
ability
    cv_scores = cross_val_score(clf, X_scaled, y, cv=5,
scoring='accuracy')

    # Print the mean cross-validation score
    print(f"Mean Cross-Validation Accuracy: {cv_scores.mean():.2f}")

Evaluating Logistic Regression with cross-validation...

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
```

```
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic
.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

Mean Cross-Validation Accuracy: 0.96
Evaluating Decision Tree with cross-validation...
Mean Cross-Validation Accuracy: 0.91
Evaluating Support Vector Machine with cross-validation...
Mean Cross-Validation Accuracy: 0.92
Evaluating Random Forest with cross-validation...
Mean Cross-Validation Accuracy: 0.96
Evaluating Gradient Boosting with cross-validation...
Mean Cross-Validation Accuracy: 0.95
```

```python
import matplotlib.pyplot as plt

# Initialize lists to store evaluation metrics
classifier_names = []
accuracies = []
precisions = []
recalls = []
f1_scores = []

# Model Training and Evaluation
for name, clf in classifiers.items():
    print(f"Training and evaluating {name}...")
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)

    # Evaluation Metrics
    accuracy = accuracy_score(y_test, predictions)
    precision = precision_score(y_test, predictions)
    recall = recall_score(y_test, predictions)
    f1 = f1_score(y_test, predictions)

    # Append metrics to lists
    classifier_names.append(name)
    accuracies.append(accuracy)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)

# Plotting
plt.figure(figsize=(10, 6))

# Accuracy
plt.plot(classifier_names, accuracies, marker='o', label='Accuracy')
```

```python
# Precision
plt.plot(classifier_names, precisions, marker='o', label='Precision')

# Recall
plt.plot(classifier_names, recalls, marker='o', label='Recall')

# F1-score
plt.plot(classifier_names, f1_scores, marker='o', label='F1-Score')

# Labels and Title
plt.xlabel('Classifier')
plt.ylabel('Score')
plt.title('Model Evaluation Metrics')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)

# Show plot
plt.tight_layout()
plt.show()
```

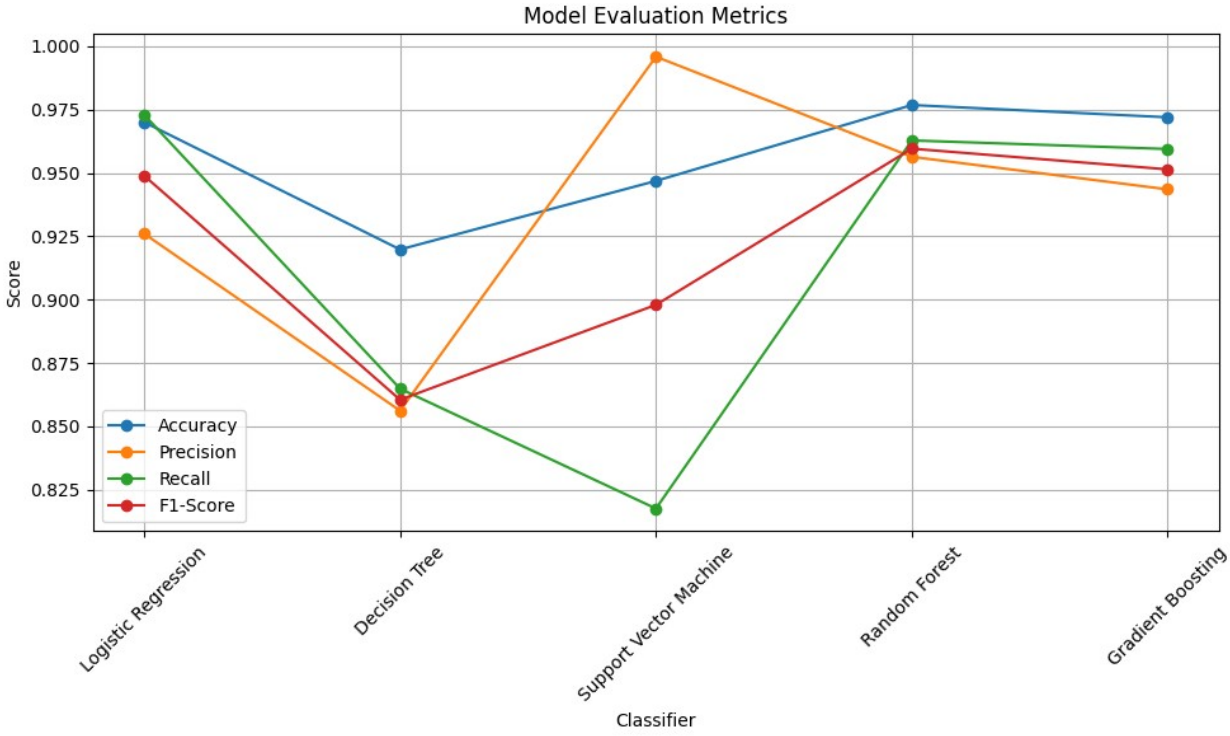
```
Training and evaluating Logistic Regression...

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

Training and evaluating Decision Tree...
Training and evaluating Support Vector Machine...
Training and evaluating Random Forest...
Training and evaluating Gradient Boosting...
```


Model Evaluation Metrics
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Score
Accuracy
Precision
Recall
F1-Score
Logistic Regression
Decision Tree
Support Vector Machine
Random Forest
Gradient Boosting
Classifier