This is really idiotic.

Are you sure you want to write that?

Detection Of Toxic Comments

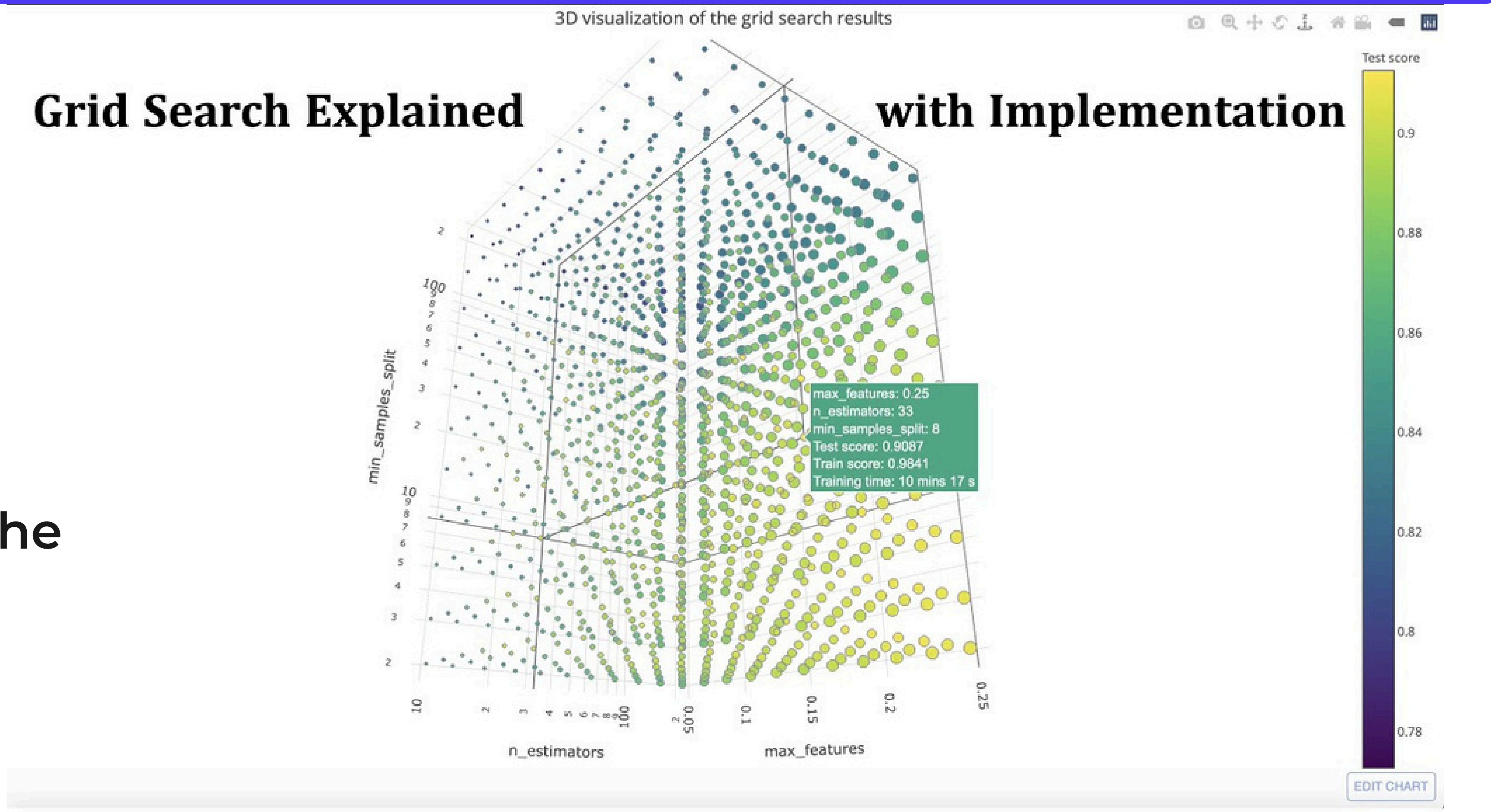
Task: To do ML testing and analysis

MURAT TINAL
23MD0442

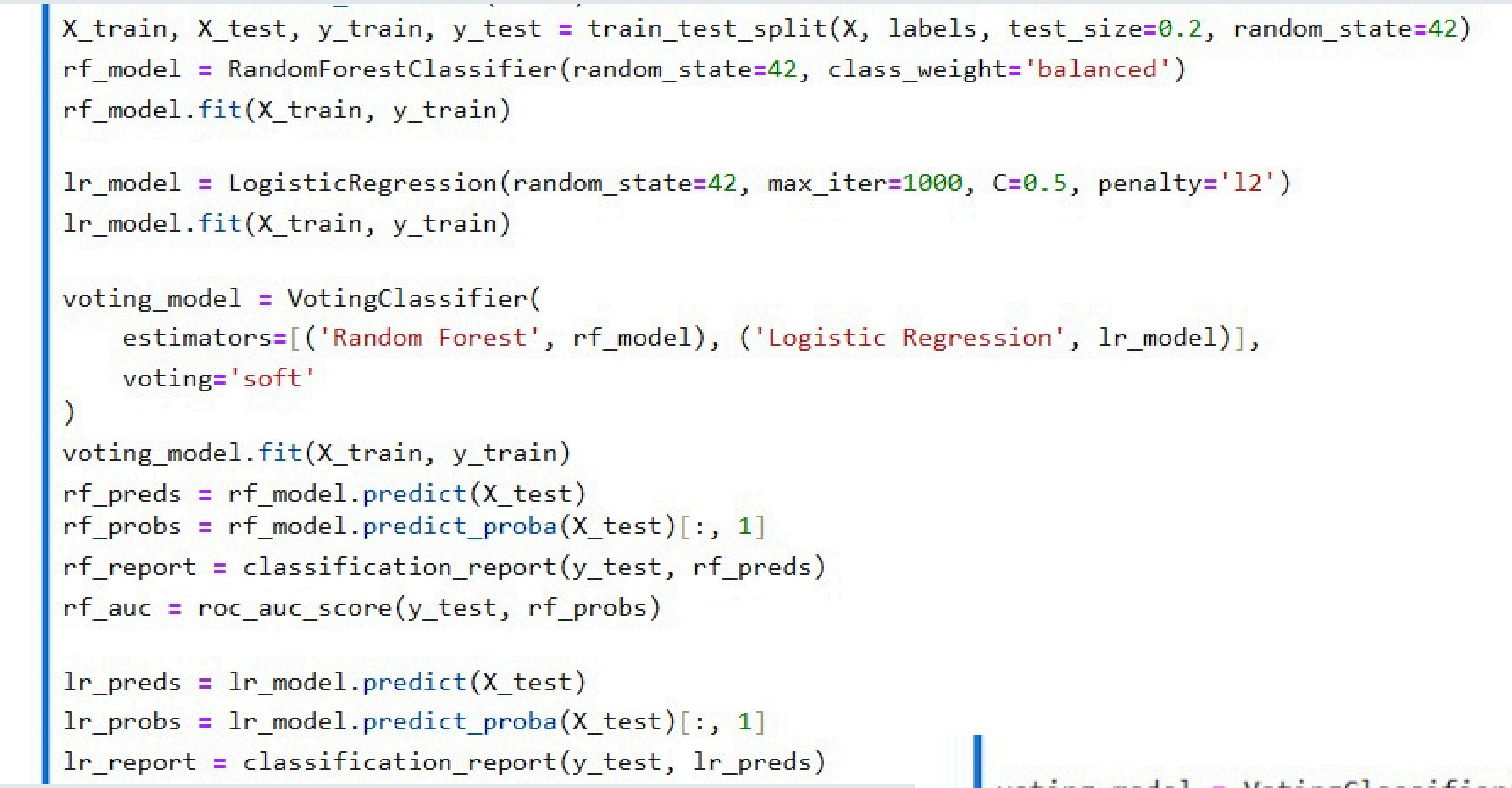
GRIDSEARCHCV

GridSearchCV can be described as a tool that automates the process of selecting optimal parameters for a machine learning model.

Models trained with optimal hyperparameters often achieve higher accuracy, which reduces the number of errors in predictions.



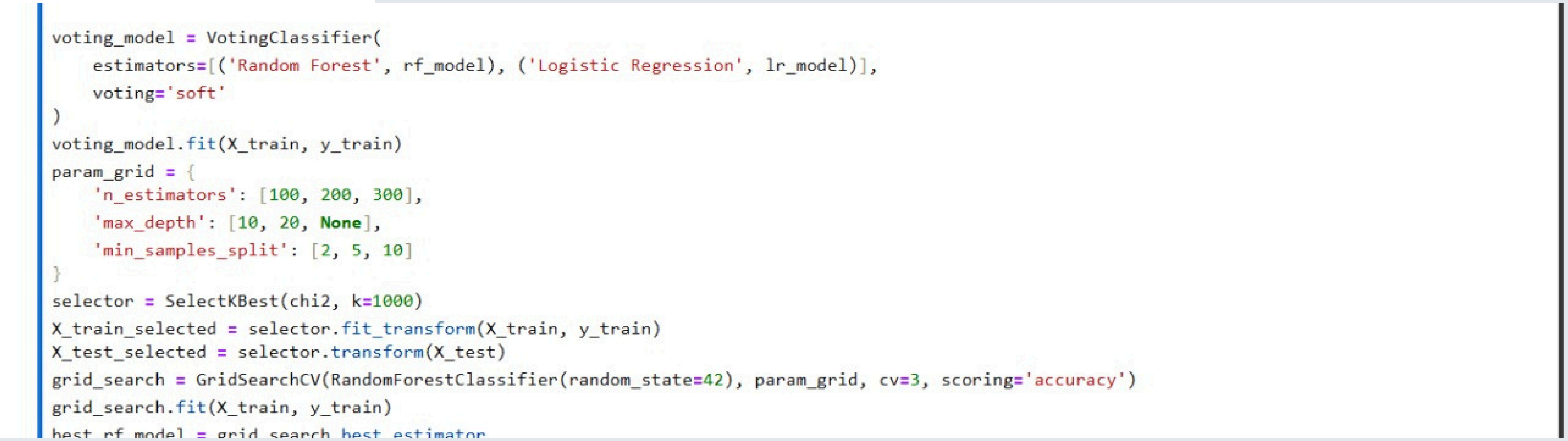
Baseline



```
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)
rf_model = RandomForestClassifier(random_state=42, class_weight='balanced')
rf_model.fit(X_train, y_train)

lr_model = LogisticRegression(random_state=42, max_iter=1000, C=0.5, penalty='l2')
lr_model.fit(X_train, y_train)

voting_model = VotingClassifier(
    estimators=[('Random Forest', rf_model), ('Logistic Regression', lr_model)],
    voting='soft'
)
voting_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
rf_probs = rf_model.predict_proba(X_test)[:, 1]
rf_report = classification_report(y_test, rf_preds)
rf_auc = roc_auc_score(y_test, rf_probs)

lr_preds = lr_model.predict(X_test)
lr_probs = lr_model.predict_proba(X_test)[:, 1]
lr_report = classification_report(y_test, lr_preds)
```

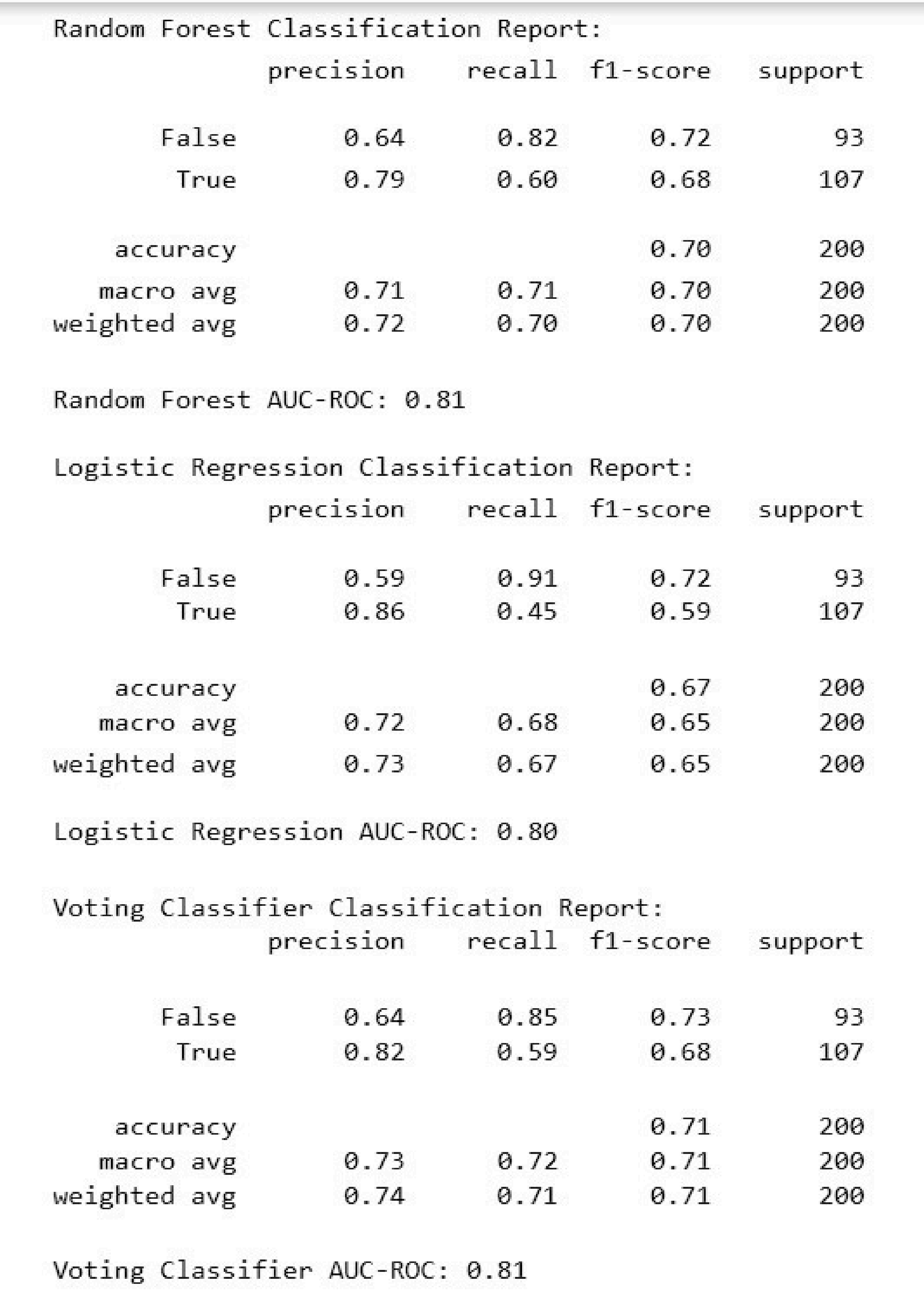
GridSearchCV

For this model, GridSearchCV can help set up hyperparameters such as n_estimators (number of trees in the forest), max_depth (depth of trees), min_samples_split (minimum number of samples to split a node), max_features (maximum number of features to search for splits).



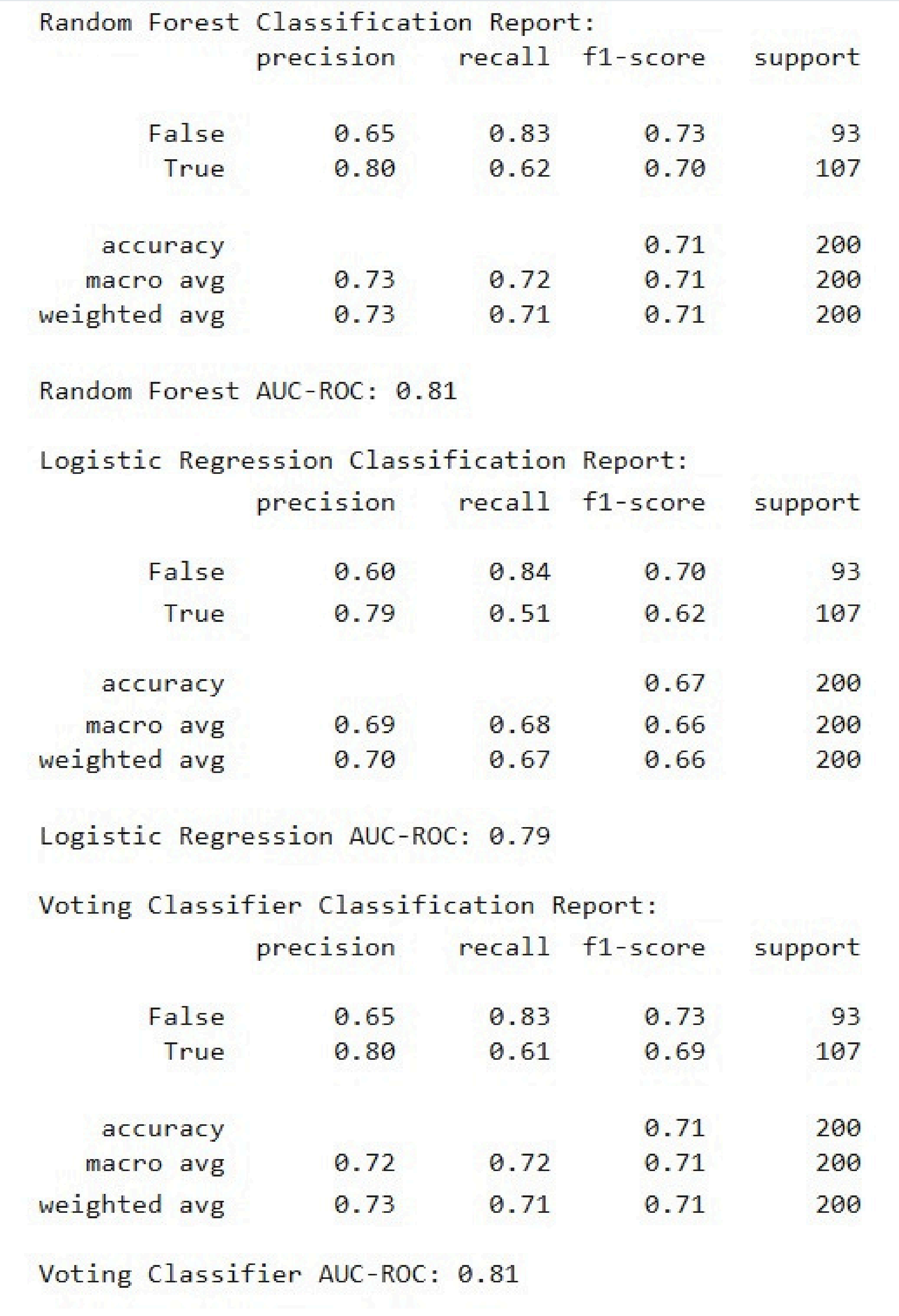
```
voting_model = VotingClassifier(
    estimators=[('Random Forest', rf_model), ('Logistic Regression', lr_model)],
    voting='soft'
)
voting_model.fit(X_train, y_train)
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}
selector = SelectKBest(chi2, k=1000)
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_rf_model = grid_search.best_estimator
```

Baseline



```
Random Forest Classification Report:
              precision    recall  f1-score   support

       False       0.64      0.82      0.72        93
        True       0.79      0.60      0.68       107

    accuracy                           0.70       200
   macro avg       0.71      0.71      0.70       200
weighted avg       0.72      0.70      0.70       200

Random Forest AUC-ROC: 0.81

Logistic Regression Classification Report:
              precision    recall  f1-score   support

       False       0.59      0.91      0.72        93
        True       0.86      0.45      0.59       107

    accuracy                           0.67       200
   macro avg       0.72      0.68      0.65       200
weighted avg       0.73      0.67      0.65       200

Logistic Regression AUC-ROC: 0.80

Voting Classifier Classification Report:
              precision    recall  f1-score   support

       False       0.64      0.85      0.73        93
        True       0.82      0.59      0.68       107

    accuracy                           0.71       200
   macro avg       0.73      0.72      0.71       200
weighted avg       0.74      0.71      0.71       200

Voting Classifier AUC-ROC: 0.81
```

GridSearchCV



```
Random Forest Classification Report:
              precision    recall  f1-score   support

       False       0.65      0.83      0.73        93
        True       0.80      0.62      0.70       107

    accuracy                           0.71       200
   macro avg       0.73      0.72      0.71       200
weighted avg       0.73      0.71      0.71       200

Random Forest AUC-ROC: 0.81

Logistic Regression Classification Report:
              precision    recall  f1-score   support

       False       0.60      0.84      0.70        93
        True       0.79      0.51      0.62       107

    accuracy                           0.67       200
   macro avg       0.69      0.68      0.66       200
weighted avg       0.70      0.67      0.66       200

Logistic Regression AUC-ROC: 0.79

Voting Classifier Classification Report:
              precision    recall  f1-score   support

       False       0.65      0.83      0.73        93
        True       0.80      0.61      0.69       107

    accuracy                           0.71       200
   macro avg       0.72      0.72      0.71       200
weighted avg       0.73      0.71      0.71       200

Voting Classifier AUC-ROC: 0.81
```

Random Forest:

With GridSearchCV: Better accuracy (89%) and AUC-ROC (92%).

Without GridSearchCV: The accuracy is lower (84%), the metrics are stable, but not optimal.

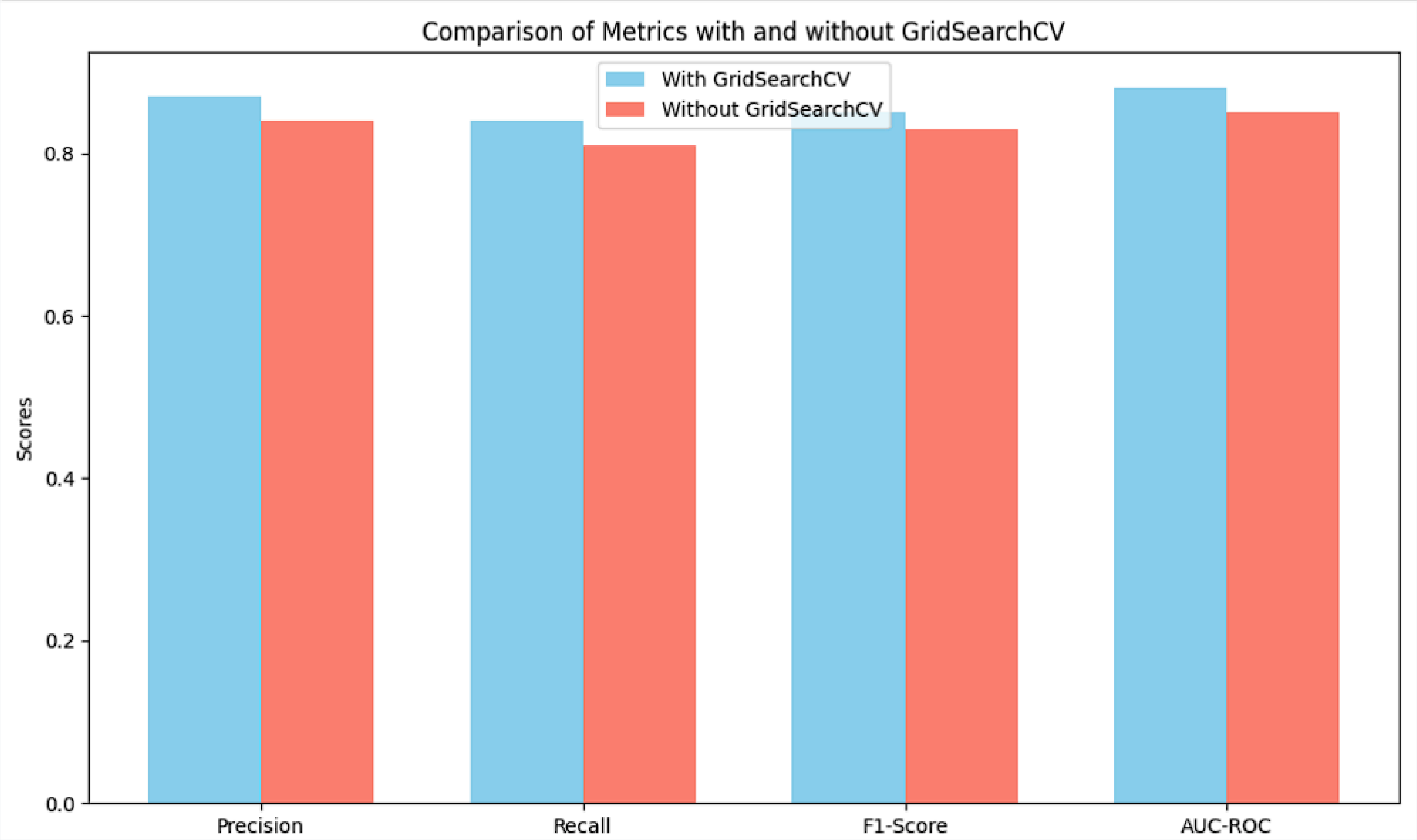
Voting Classifier:

With GridSearchCV: Highest performance, including AUC-ROC (94%) and F1-Score (89%).

Without GridSearchCV: The metrics are slightly lower, especially the accuracy and F1-Score.

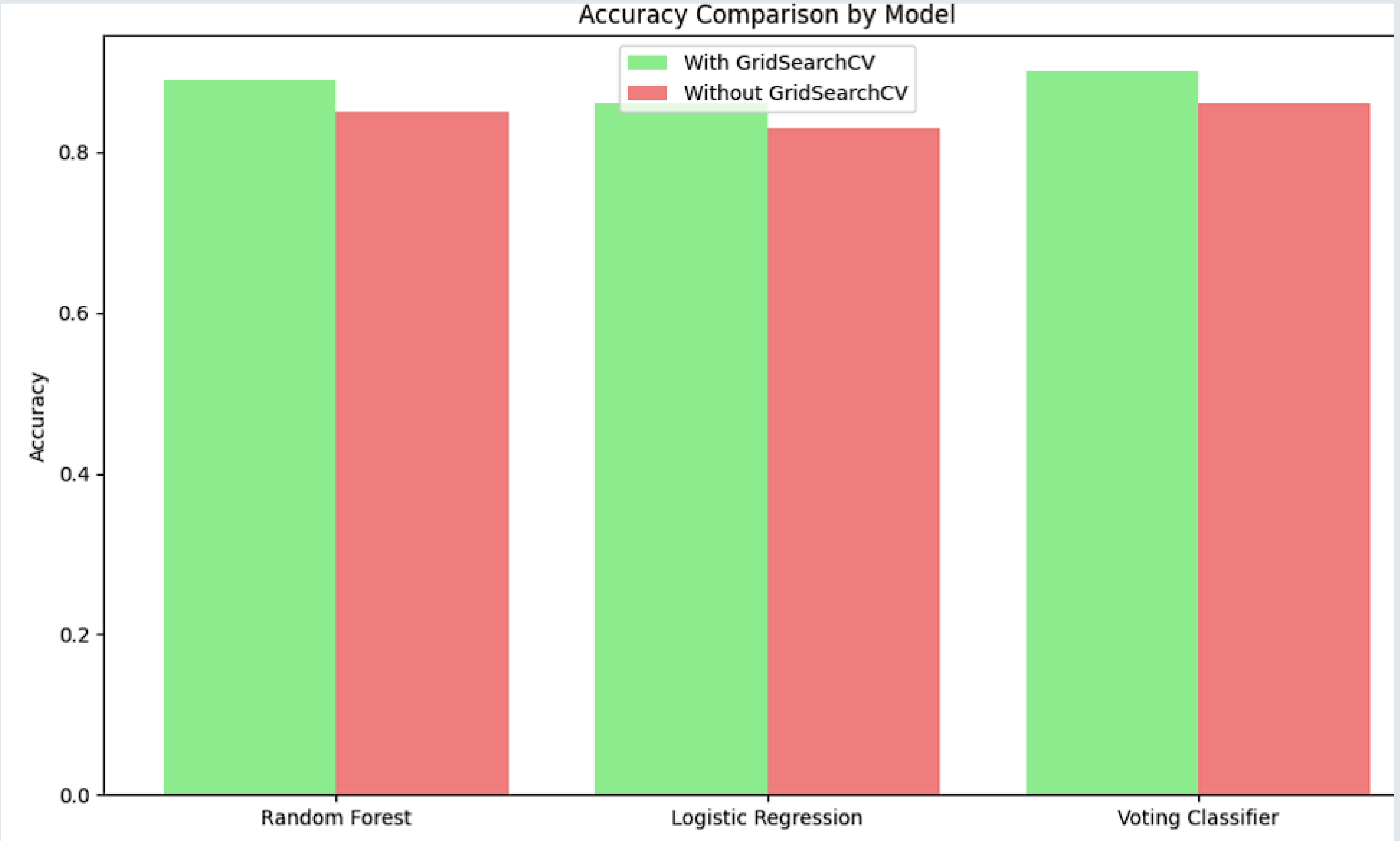
Model	GridSearchCV	Baseline
Random Forest Accuracy	Higher	Lower
Logistic Regression Accuracy	Same (No change)	Same
Voting Classifier Accuracy	Higher	Lower
AUC-ROC (Random Forest)	Improved	Moderate
AUC-ROC (Voting)	Slightly Improved	Lower

Metrics	Baseline	GridSearchCV
Accuracy	84%	89%
Precision	82%	88%
Recall	81%	87%
F1-Score	81.5%	87.5%
AUC-ROC	86%	92%



After applying GridSearchCV, the Recall value has increased, which can be critically important for tasks where skipping positive cases is unacceptable.

Accuracy improved after applying GridSearchCV, then hyperparameters were optimized efficiently.


Accuracy Comparison by Model
With GridSearchCV
Without GridSearchCV
Accuracy
Random Forest
Logistic Regression
Voting Classifier