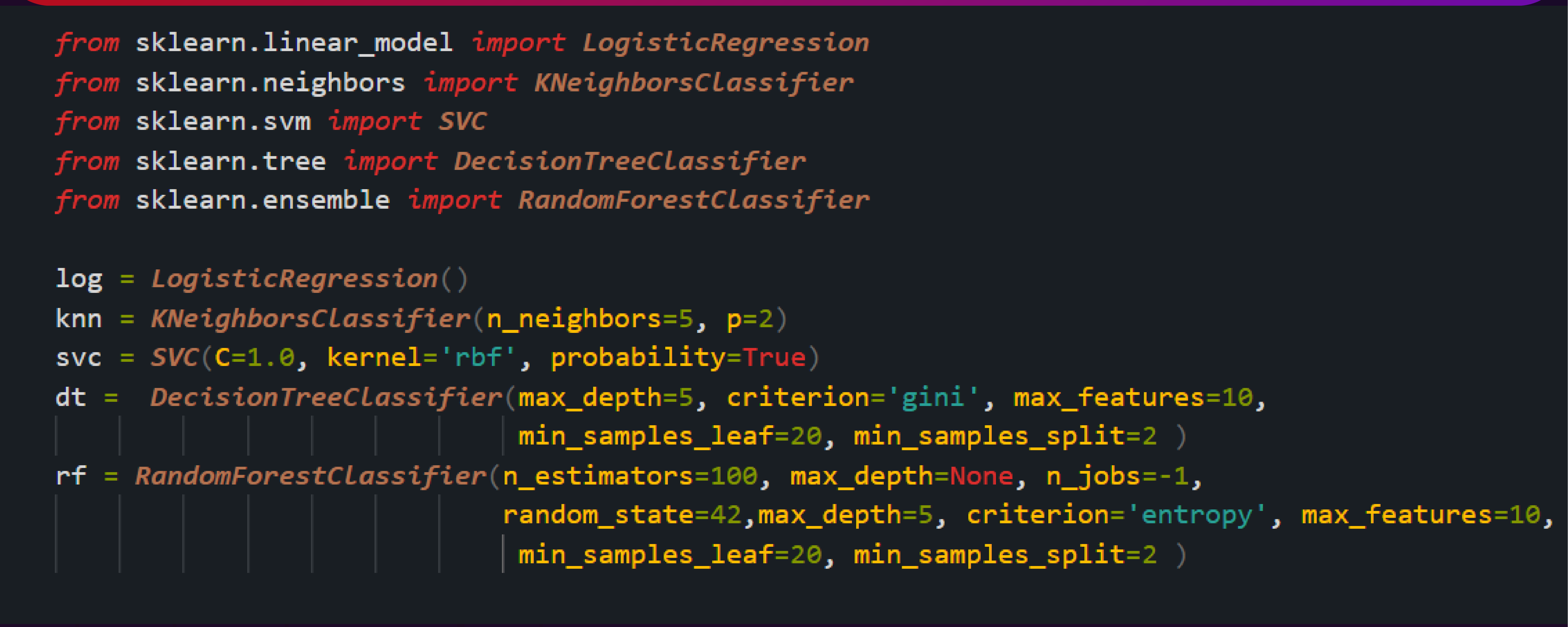
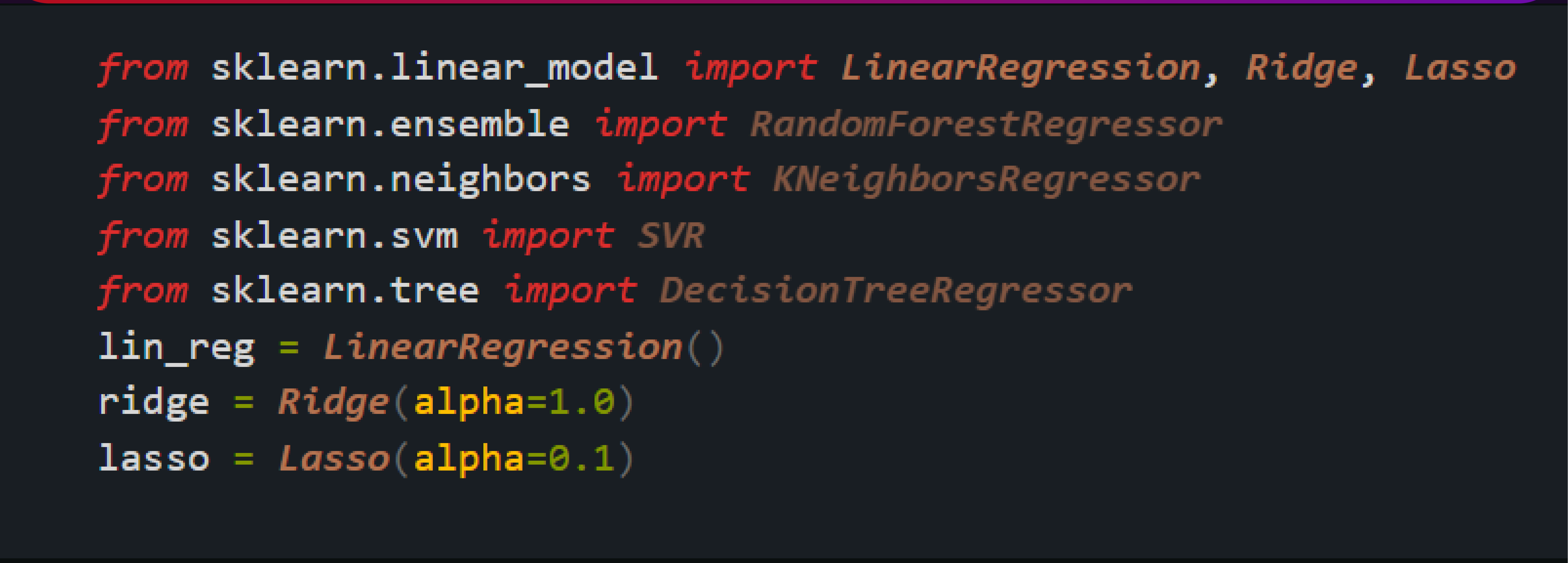
SCIKIT-LEARN CHEAT SHEET

CLASSIFICATION MODELS



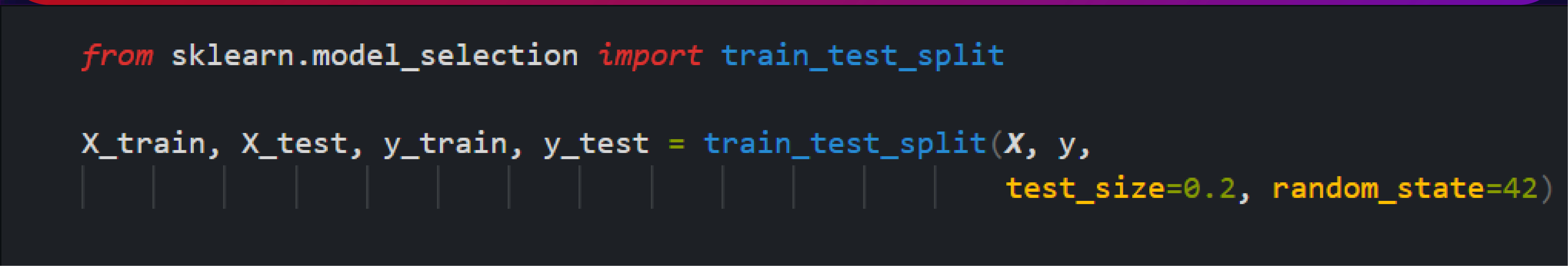
```python
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

log = LogisticRegression()
knn = KNeighborsClassifier(n_neighbors=5, p=2)
svc = SVC(C=1.0, kernel='rbf', probability=True)
dt =  DecisionTreeClassifier(max_depth=5, criterion='gini', max_features=10,
                              min_samples_leaf=20, min_samples_split=2 )
rf = RandomForestClassifier(n_estimators=100, max_depth=None, n_jobs=-1,
                            random_state=42,max_depth=5, criterion='entropy', max_features=10,
                             min_samples_leaf=20, min_samples_split=2 )
```

REGRESSION MODELS



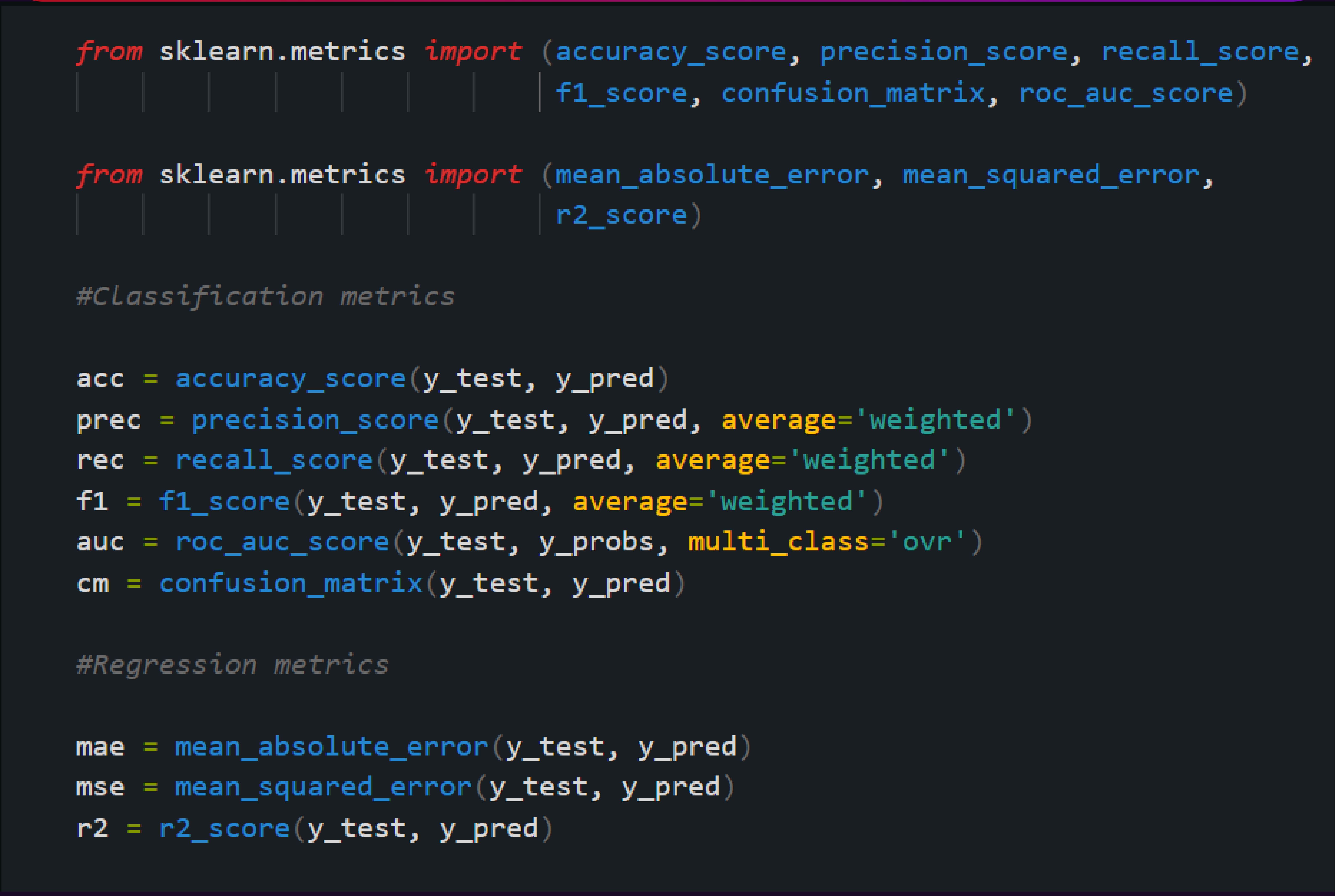
```python
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
lin_reg = LinearRegression()
ridge = Ridge(alpha=1.0)
lasso = Lasso(alpha=0.1)
```

TRAIN TEST SPLIT

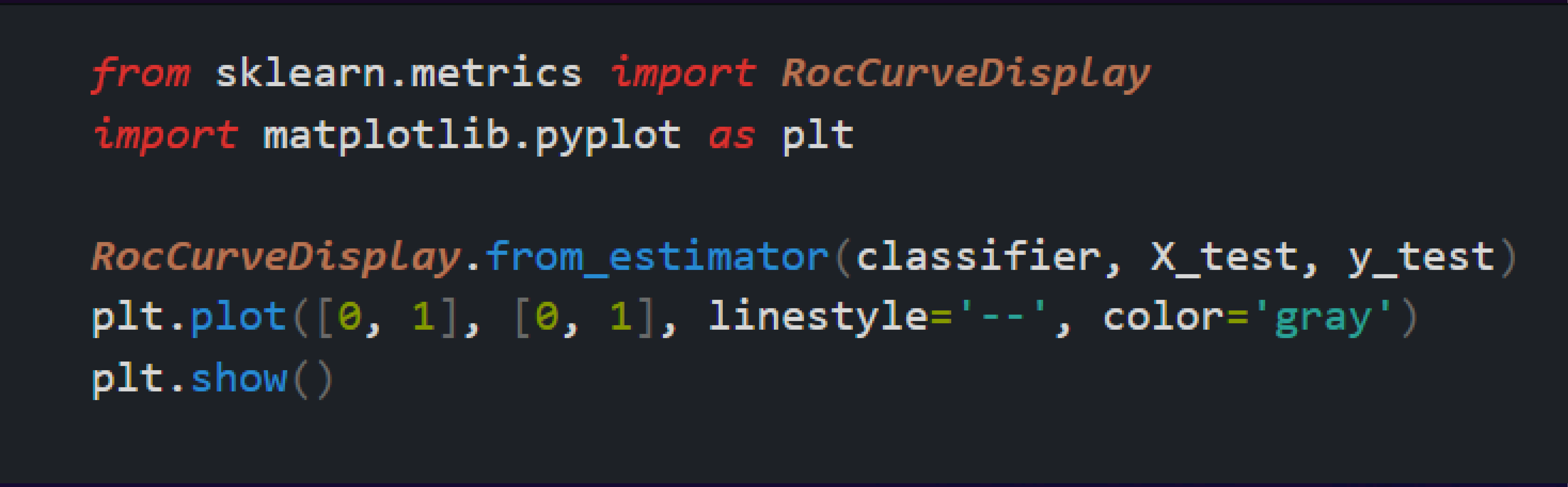


```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=42)
```

EVALUATION METRICS

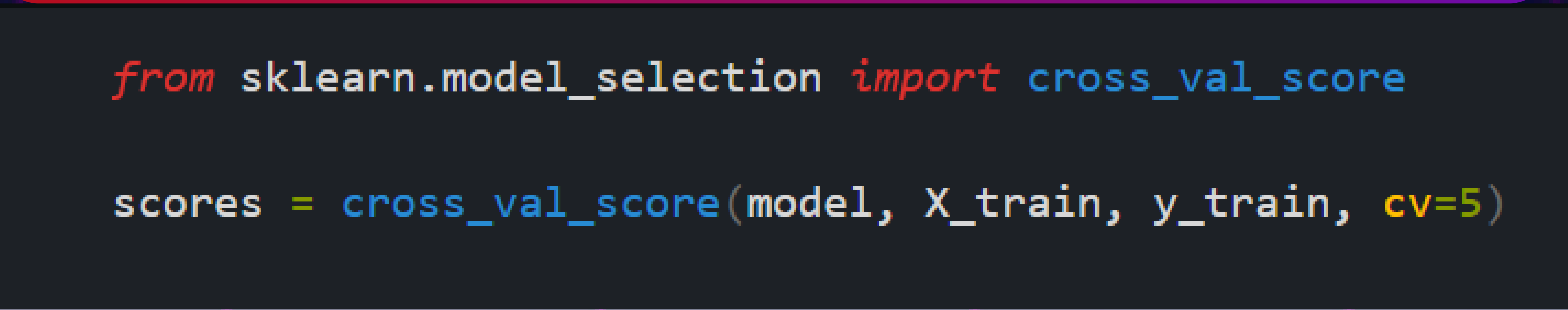


```
from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                             f1_score, confusion_matrix, roc_auc_score)

from sklearn.metrics import (mean_absolute_error, mean_squared_error,
                             r2_score)

#Classification metrics

acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='weighted')
rec = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
auc = roc_auc_score(y_test, y_probs, multi_class='ovr')
cm = confusion_matrix(y_test, y_pred)

#Regression metrics

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```



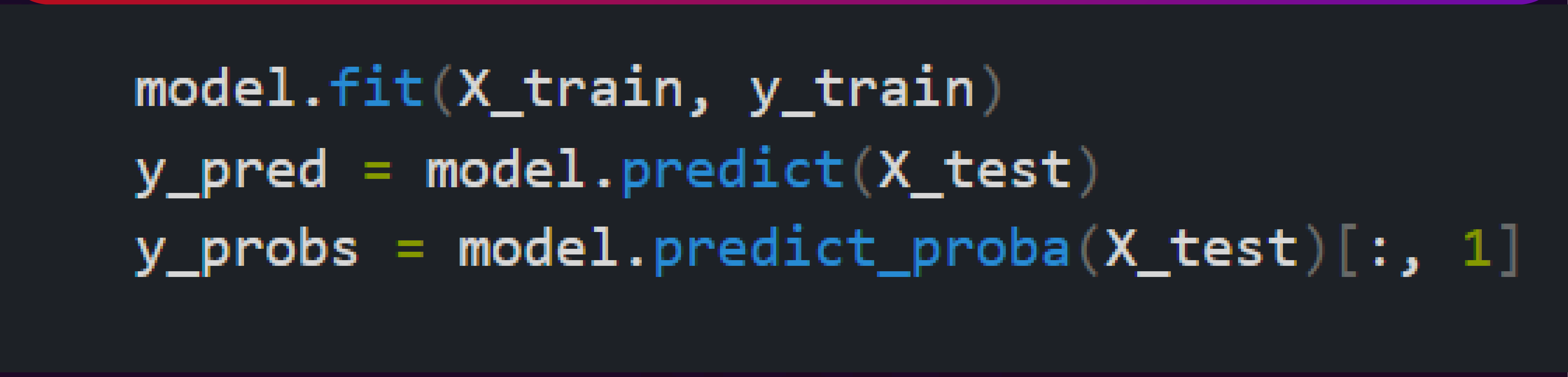
```
from sklearn.metrics import RocCurveDisplay
import matplotlib.pyplot as plt

RocCurveDisplay.from_estimator(classifier, X_test, y_test)
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.show()
```

CROSS VALIDATION



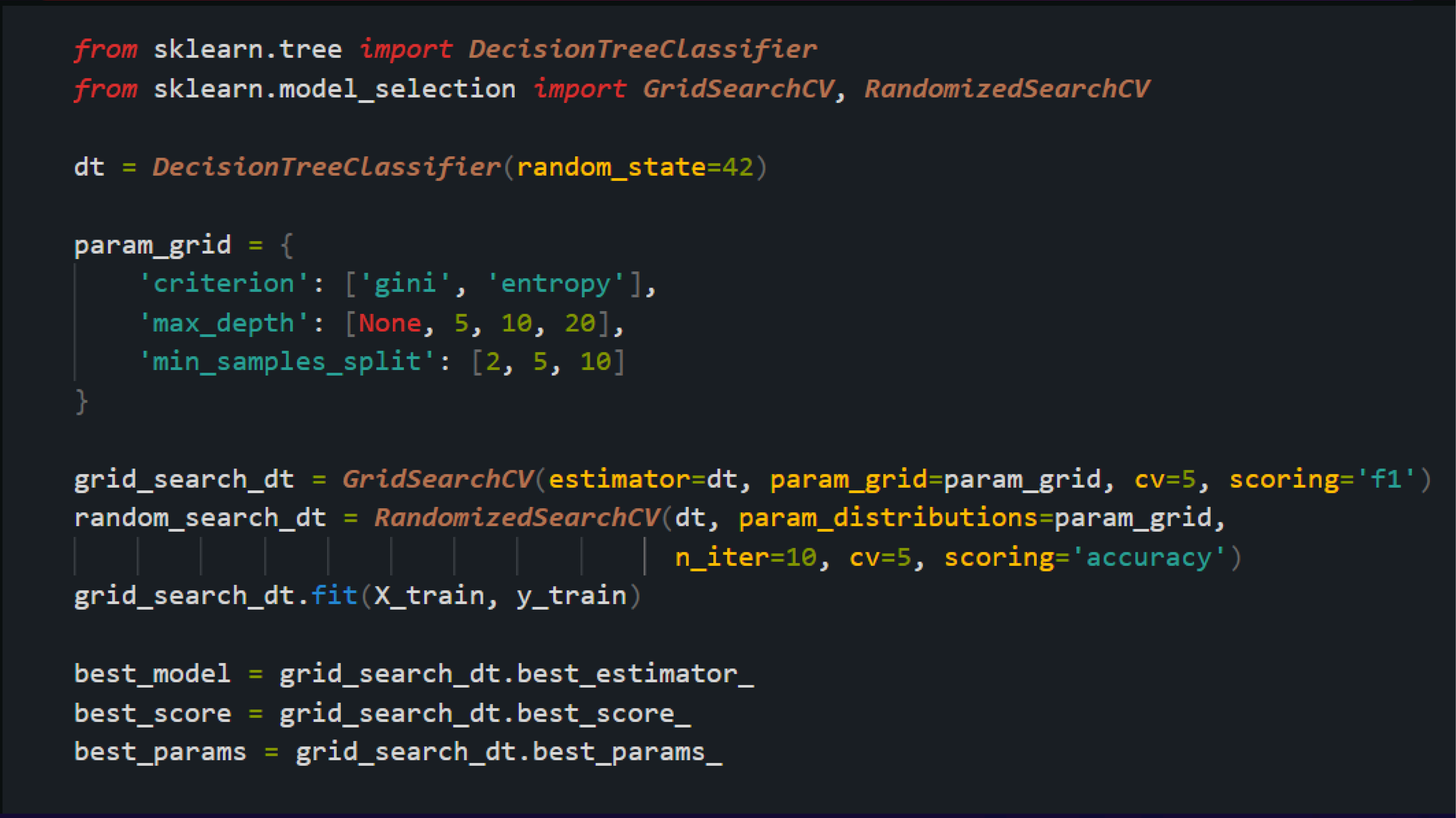
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X_train, y_train, cv=5)
```

TRAINING/MAKING PREDICTIONS



```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
y_probs = model.predict_proba(X_test)[:, 1]
```

FINE-TUNING



```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

dt = DecisionTreeClassifier(random_state=42)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10]
}

grid_search_dt = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, scoring='f1')
random_search_dt = RandomizedSearchCV(dt, param_distributions=param_grid,
                                      n_iter=10, cv=5, scoring='accuracy')
grid_search_dt.fit(X_train, y_train)

best_model = grid_search_dt.best_estimator_
best_score = grid_search_dt.best_score_
best_params = grid_search_dt.best_params_
```

NeuroDynamics

Where Brains And Machines Converge