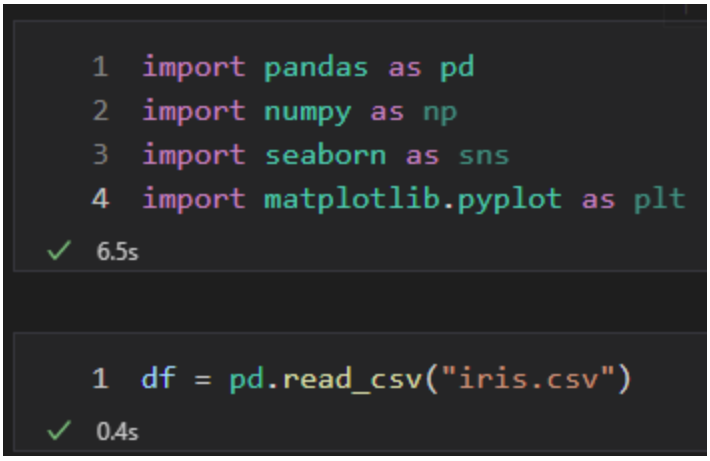
Multiclass Regr



▼ Model

- imports, prepare and train test split

```
1  X = df.drop("species", axis=1)
2  y = df["species"]
```
✓ 0.3s

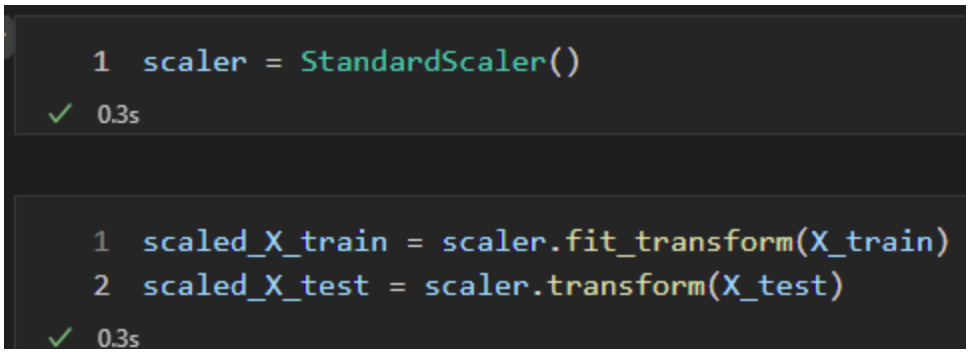
```
1  y
```
✓ 0.3s

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
        ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

```
1  from sklearn.model_selection import train_test_split
2  from sklearn.preprocessing import StandardScaler
```
✓ 0.8s

```
1  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=101)
```
✓ 0.4s

- Scaler Model



- Regression tool import
  Penalty, l1 and C values
  Param grid
  grid model

```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.model_selection import GridSearchCV
```
✓ 0.5s

```
1 log_model = LogisticRegression(solver="saga",multi_class="ovr", max_iter=5000)
```
✓ 0.5s

```
1 penalty = ["l1","l2","elasticnet"]
2 l1_ratio = np.linspace(0,1,20)
3 C = np.logspace(0,10,20)
4
5 param_grid={
6     "penalty" : penalty,
7     "l1_ratio" : l1_ratio,
8     "C" : C
9 }
```
✓ 0.3s

```
1 grid_model = GridSearchCV(log_model, param_grid=param_grid)
```
✓ 0.4s

- grid model fit

```
1 grid_model.fit(scaled_X_train, y_train)
```
✓ 3m 36.9s

```
Output exceeds the size limit. Open the full output data in a text editor
C:\Users\mbatu\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:1317: UserWarning: l1_ratio
parameter is only used when penalty is 'elasticnet'. Got (penalty=l1)
  warnings.warn("l1_ratio parameter is only used when penalty is "
C:\Users\mbatu\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:1317: UserWarning: l1_ratio
parameter is only used when penalty is 'elasticnet'. Got (penalty=l1)
  warnings.warn("l1_ratio parameter is only used when penalty is "
C:\Users\mbatu\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:1317: UserWarning: l1_ratio
parameter is only used when penalty is 'elasticnet'. Got (penalty=l1)
```

- Best Parameters

```
1  from sklearn.metrics import accuracy_score, confusion_matrix
2  from sklearn.metrics import classification_report, plot_confusion_matrix
```
✓ 0.3s

```
1  grid_model.best_params_
```
✓ 0.6s

```
{'C': 11.28837891684689, 'l1_ratio': 0.0, 'penalty': 'l1'}
```

- predictions

```
1  y_pred = grid_model.predict(scaled_X_test)
2  y_pred
```
0.6s

```
ray(['setosa', 'setosa', 'setosa', 'virginica', 'versicolor',
    'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa',
    'virginica', 'setosa', 'setosa', 'virginica', 'virginica',
    'versicolor', 'versicolor', 'versicolor', 'setosa', 'versicolor',
    'versicolor', 'setosa', 'versicolor', 'versicolor', 'versicolor',
    'versicolor', 'versicolor', 'virginica', 'setosa', 'setosa',
    'virginica', 'versicolor', 'virginica', 'versicolor', 'virginica',
    'versicolor', 'versicolor', 'versicolor'], dtype=object)
```

- accuracy score, confusion matrix and its plot

```
1  accuracy_score(y_test,y_pred)
```
✓ 0.9s

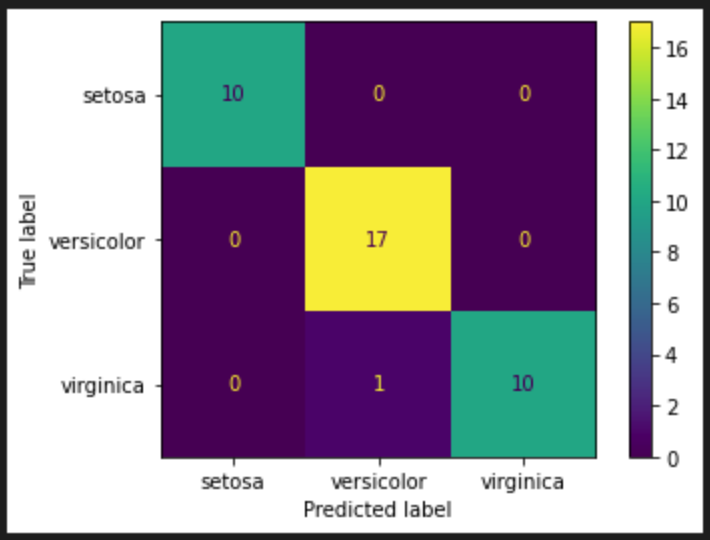
```
0.9736842105263158
```

```
1  confusion_matrix(y_test,y_pred)
```
✓ 0.8s

```
array([[10,  0,  0],
       [ 0, 17,  0],
       [ 0,  1, 10]], dtype=int64)
```

```
1  plot_confusion_matrix(grid_model, scaled_X_test,y_test)
```
✓ 0.4s

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay
```



- classification report

```
1  print(classification_report(y_test,y_pred))
✓ 0.8s
```

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	10
versicolor	0.94	1.00	0.97	17
virginica	1.00	0.91	0.95	11
accuracy			0.97	38
macro avg	0.98	0.97	0.97	38
weighted avg	0.98	0.97	0.97	38

- imports and plot multi label ROC plot

```python
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import roc_curve, auc
```

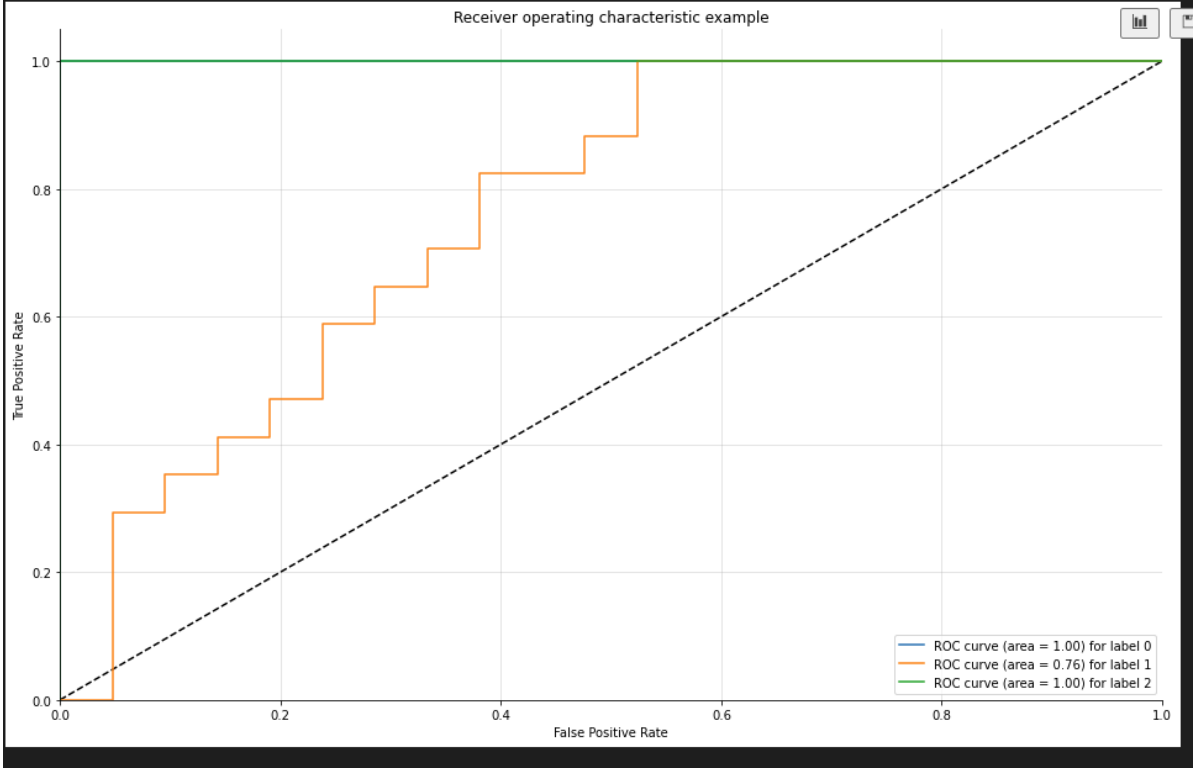
✓ 0.7s

```python
def plot_multiclass_roc(clf, X_test, y_test, n_classes, figsize=(5,5)):
    y_score = clf.decision_function(X_test)

    # structures
    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    # calculate dummies once
    y_test_dummies = pd.get_dummies(y_test, drop_first=False).values
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_dummies[:, i], y_score[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # roc for each class
    fig, ax = plt.subplots(figsize=figsize)
    ax.plot([0, 1], [0, 1], 'k--')
    ax.set_xlim([0.0, 1.0])
    ax.set_ylim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.set_title('Receiver operating characteristic example')
    for i in range(n_classes):
        ax.plot(fpr[i], tpr[i], label='ROC curve (area = %0.2f) for label %i' % (roc_auc[i], i))
    ax.legend(loc="best")
    ax.grid(alpha=.4)
    sns.despine()
    plt.show()
```

✓ 0.1s

- ROC plot

```
1  plot_multiclass_roc(grid_model, scaled_X_test, y_test, n_classes=3, figsize=(16, 10))
```
✓ 0.3s



▼ Heart Disease

- logistic regression cv

```
1  from sklearn.linear_model import LogisticRegressionCV
✓ 0.7s

1  log_model = LogisticRegressionCV()
✓ 0.4s

1  log_model.fit(scaled_X_train,y_train)
✓ 0.3s
LogisticRegressionCV()

1  log_model.Cs_
✓ 0.1s
array([1.00000000e-04, 7.74263683e-04, 5.99484250e-03, 4.64158883e-02,
       3.59381366e-01, 2.78255940e+00, 2.15443469e+01, 1.66810054e+02,
       1.29154967e+03, 1.00000000e+04])
```

- coefficients
  data=log_model.coef_[0] : içiçe 2 seri var içerdekini almak için [0] kullanıldı.

```
1 log_model.coef_
```
✓ 0.6s

```
array([[-0.09621199, -0.39460154,  0.53534731, -0.13850191, -0.08830462,
         0.02487341,  0.08083826,  0.29914053, -0.33438151, -0.352386  ,
         0.25101033, -0.49735752, -0.37448551]])
```

Coefficient Plot

```
1 coefs = pd.Series(index=X.columns, data=log_model.coef_[0])
2 coefs
```
✓ 0.6s

```
age        -0.096212
sex        -0.394602
cp          0.535347
trestbps   -0.138502
chol       -0.088305
fbs         0.024873
restecg     0.080838
thalach     0.299141
exang      -0.334382
oldpeak    -0.352386
slope       0.251010
ca         -0.497358
thal       -0.374486
dtype: float64
```

- coef bar plot
  coefs = coefs.sort_values() : değerleri küçükten büyüğe sıralar ve coefs yerine yazar. kalıcı.

```
1  plt.figure(figsize=(12,8),dpi=200)
2  coefs = coefs.sort_values()
3  sns.barplot(x=coefs.index, y=coefs.values)
```

✓ 0.4s Python

```
<AxesSubplot:>
```