Confusion Matrix Code

```
classes = np.unique(df['Column_name'])
matrix = np.zeros((len(classes), len(classes)))
for i in range(len(classes)):
    for j in range(len(classes)):
        matrix[i,j] = np.sum((actual==classes[i]) &
                             (predicted==classes[j]))
```

$FP = matrix.sum(axis=0) - np.diag(matrix)$

$FN = matrix.sum(axis=1) - np.diag(matrix)$

$TP = np.diag(matrix)$

$TN = matrix.sum() - (FP + FN + TP)$

$TPR = TP / TP + FN$

$TNR = TN / TN + FP$

$FPR = FP / FP + TN$

$FNR = FN / TP + FN$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$F1\_score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$(+)\ Precision = \frac{TP}{TP + FP} \qquad (-)\ Precision = \frac{TN}{TN + FN}$$

$$(+)\ Recall = \frac{TP}{TP + FN} \qquad (-)\ Recall = \frac{TN}{TN + FP}$$

$$F1\_\beta = \frac{(1+\beta^2) * prec * recall}{(\beta^2 * prec) + recall}$$

Normalize

Min-Max formula

df['Colum'] = (df['column'] - df['column'].min()) / (df['column'].max() - df['column'].min())

PCA

features = ds.T (Transpose)

cov_matrix = np.cov(features) [Covariance Matrix]

values, vectors = np.linalg.eig(cov_matrix)

Calculate variance

variance = []
for i in range(len(values)):
    variances.append(values[i] / np.sum(values))

pc1 = ds.dot(vectors.T[0])
pc1
pc2 = ds.dot(vectors.T[1])
pc2

Evaluation Parameters

mean square error (MSE)

$$= \sum_{i=1}^{n} \frac{(\text{predicted} - \text{actual})}{n}$$

root mean square (RMSE) = $\sqrt{MSE}$.

Mean Absolute Error = $\frac{1}{n} \sum_{i=1}^{n} |x - x_i|$ → mean.

Correlation.

```
y_true = ds['actual']
y_pred = ds['predicted']
correlation = y_true.corr[y_pred]
```

R^2 score

$$R^2 = (\text{Correlation})^2$$

Inbuilt

```
from sklearn.metrics import r2_score
r2 = r2_score(y_true, y_pred)
r2.
```