Sci-Kit learn Cheat Sheet

by saicharan via cheatography.com/121400/cs/22200/

loading the data into the model

```
>>> import numpy as np
>>> X = np.random.random((9,6))
>>> y = np.array(['M','M','m','f','m','m','f','-M','F','na','m'])
>>> X[X < =0.2] = 0
```

Estimators used in unsupervised learning models

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

```
>>> k_means.fit(X_train)
>>> pca_model = pca.fit_transform(X_train)
Prediction
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2])
>>> mean_absolute_error(y_true, y_pred))
```

Data pre-processing

Standardization technique

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization technique

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Attributing Missing Values

```
>>>from sklearn.preprocessing import Imputer
>>>imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>>imp.fit_transform(X_train)
```

Performance metrics

Accuracy Score

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred)))
```

Model Tuning

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3), "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Estimators used in supervised learning models

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

K- nearest neighbors

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```