Scikit-learn Beginner Cheat Sheet

1. Standard Imports

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
    mean_squared_error
```

2. The Scikit-learn Workflow

The basic steps for any supervised learning project.

1. **Load Data**: Get your data into a NumPy array or Pandas DataFrame.
2. **Split Data**: Separate data into features (**X**) and the target variable (**y**).
3. **Train-Test Split**: Divide X and y into training and testing sets.
4. **Choose an Estimator**: Select a model (e.g., 'LinearRegression', 'LogisticRegression').
5. **Fit the Model**: Train the model on your training data using `.fit()`.
6. **Predict**: Make predictions on the test data using `.predict()`.
7. **Evaluate**: Compare the predictions to the actual test labels to measure performance.

3. Data Preparation

Getting your data ready for the model.

Loading Sample Data

Scikit-learn comes with sample datasets.

```
from sklearn.datasets import load_iris
iris = load_iris()
X, y = iris.data, iris.target
# X is a NumPy array of features
# y is a NumPy array of target labels
```

Train-Test Split

The most important step to prevent data leakage.

```
# Split data into 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# random_state ensures the split is reproducible
```

4. Preprocessing Data

Most models require data to be scaled. The pattern is `.fit()` on training data, then `.transform()` on both train and test data.

Feature Scaling ('StandardScaler')

Scales data to have a mean of 0 and a standard deviation of 1.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fit the scaler ONLY on the training data
scaler.fit(X_train)

# Transform both train and test data
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

5. Supervised Learning: Classification

Predicting a categorical label (e.g., 'spam' vs 'not spam').

Example: Logistic Regression

```
from sklearn.linear_model import LogisticRegression

# 1. Instantiate the model
model = LogisticRegression()

# 2. Fit the model on scaled training data
model.fit(X_train_scaled, y_train)

# 3. Make predictions on the scaled test data
y_pred = model.predict(X_test_scaled)
```

Evaluating Classification

```
from sklearn.metrics import accuracy_score,
    confusion_matrix

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Get the confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Rows: True labels, Columns: Predicted labels
print(cm)
```

6. Supervised Learning: Regression

Predicting a continuous numerical value (e.g., price, temperature).

Example: Linear Regression

```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression

# Generate synthetic regression data
X, y = make_regression(n_samples=100, n_features=1,
    noise=10)
X_train, X_test, y_train, y_test = train_test_split(
    X, y)

# 1. Instantiate and fit the model
reg_model = LinearRegression()
reg_model.fit(X_train, y_train)

# 2. Make predictions
y_pred_reg = reg_model.predict(X_test)
```

Scikit-learn Beginner Cheat Sheet

Evaluating Regression

```
from sklearn.metrics import mean_squared_error,
    r2_score

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_reg)
print(f'MSE: {mse:.2f}')

# Calculate R-squared (coefficient of determination)
r2 = r2_score(y_test, y_pred_reg)
print(f'R-squared: {r2:.2f}') # Best is 1.0
```

7. Unsupervised Learning: Clustering

Finding groups in data without pre-defined labels. Note: no y variable or train-test split is used for fitting.

Example: K-Means Clustering

```
from sklearn.cluster import KMeans

# 1. Instantiate the model
# n_clusters is the number of groups to find
kmeans = KMeans(n_clusters=3, n_init='auto')

# 2. Fit the model to the data (e.g., iris features)
kmeans.fit(X) # Using original iris data X

# 3. Get the cluster labels for each data point
labels = kmeans.labels_
print(labels)
```

8. Saving and Loading Models

Save your trained model to a file so you can use it later without retraining.

```
import joblib

# Assume 'model' is our trained LogisticRegression
    model
# Save the model to a file
joblib.dump(model, 'my_model.pkl')

# ... later, in another script ...

# Load the model from the file
loaded_model = joblib.load('my_model.pkl')

# loaded_model is now ready to make predictions
# loaded_model.predict(...)
```