Orange to Python Cheat Sheet

Use this guide to translate common Orange workflow steps into Python code using `pandas` and `scikit-learn`. Run notebooks with `uv run -with jupyter jupyter lab` or work in Google Colab and install packages using `!pip install`.

Common scikit-learn imports

Import	**Purpose**
`from sklearn.pipeline import Pipeline`	chain preprocessing and model steps
`from sklearn.compose import ColumnTransformer`	apply transforms to column subsets
`from sklearn.impute import SimpleImputer`	fill missing values
`from sklearn.preprocessing import OneHotEncoder, StandardScaler`	encode categories, scale numbers
`from sklearn.linear_model import LogisticRegression`	logistic regression classifier
`from sklearn.neighbors import KNeighborsClassifier`	k-nearest neighbours classifier

Orange Widget / Step	**Python Equivalent**

File	`import pandas as pd`
	`data = pd.read_csv("data.csv")`
	`# or pd.read_excel("data.xlsx")`

Data Table	a pandas `DataFrame` called `data`
Select Columns	`data = data[["col1", "col2"]]`
Select Rows	`data = data[data["col"] > value]`
Concatenate	`pd.concat([df1, df2])`

Impute	`from sklearn.impute import SimpleImputer`
	`data = SimpleImputer().fit_transform(data)`

Drop Missing	`data = data.dropna()`

Normalize	`from sklearn.preprocessing import StandardScaler`
	`data = StandardScaler().fit_transform(data)`

Discretize	`from sklearn.preprocessing import KBinsDiscretizer`
	`data = KBinsDiscretizer().fit_transform(data)`

One Hot	`pd.get_dummies(data)`

PCA	`from sklearn.decomposition import PCA`
	`components = PCA(n_components=2).fit_transform(data)`

t-SNE	`from sklearn.manifold import TSNE`
	`TSNE(perplexity=30).fit_transform(data)`

K Means	`from sklearn.cluster import KMeans`
	`KMeans(n_clusters=3, random_state=0).fit_predict(data)`

DBSCAN	`from sklearn.cluster import DBSCAN`
	`DBSCAN(eps=0.5, min_samples=5).fit_predict(data)`

Logistic Regression	`from sklearn.linear_model import LogisticRegression`
	`LogisticRegression().fit(X, y)`

Random Forest	`from sklearn.ensemble import RandomForestClassifier`
	`RandomForestClassifier().fit(X, y)`

Train/Test Split	`from sklearn.model_selection import train_test_split`
	`X_train, X_test, y_train, y_test = train_test_split(X, y)`

Test & Score	`from sklearn.model_selection import cross_val_score`
	`cross_val_score(model, X, y, cv=5)`

Save Model	`import joblib`
	`joblib.dump(model, "model.joblib")`

Load Model	`model = joblib.load("model.joblib")`
Predict	`model.predict(X_new)`
Scatter Plot	`data.plot.scatter(x="col1", y="col2")`
Histogram	`data["col"].hist()`
Save Data	`data.to_csv("output.csv", index=False)`

Unsupervised Sample Program

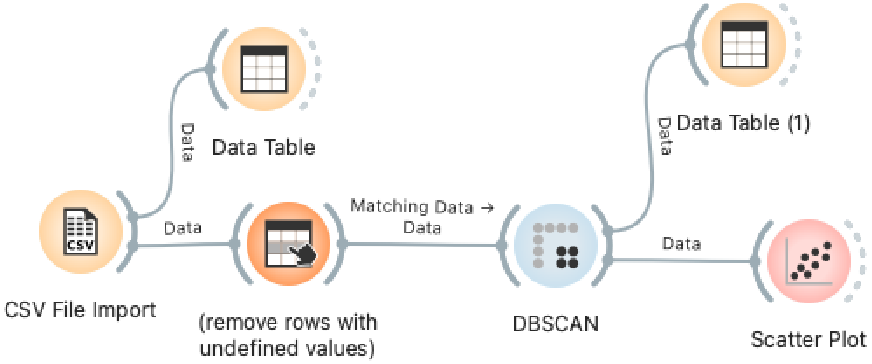

Figure 1: Clustering workflow in Orange

```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Load data
data = pd.read_csv("data.csv")

# Handle missing values
data = SimpleImputer(strategy="mean").fit_transform(data)

# Scale features
data = StandardScaler().fit_transform(data)

# Cluster
labels = KMeans(n_clusters=3, random_state=0).fit_predict(data)
print(labels)
```

Supervised Learning with Cross-Validation and Model Saving

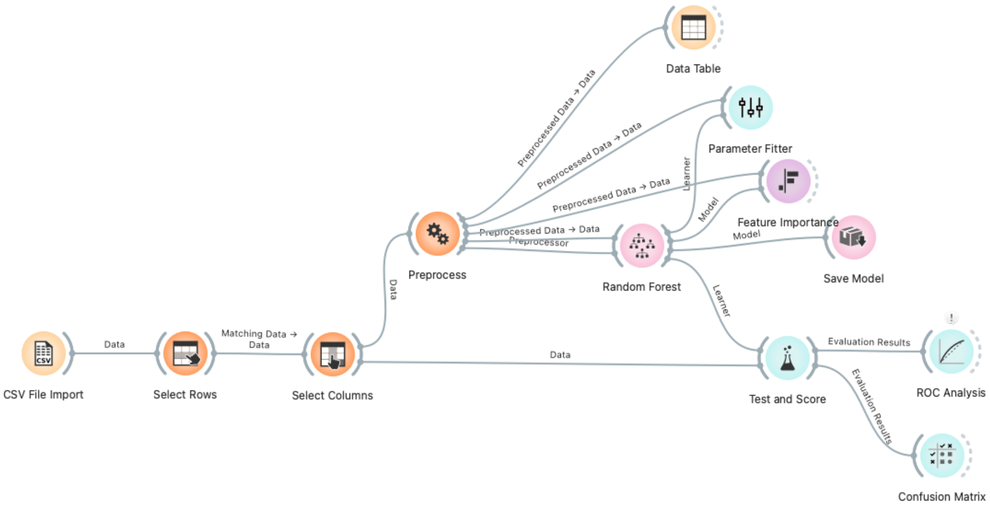

Figure 2: Random forest workflow in Orange

```
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
import joblib

# Load data with a column named 'target'
data = pd.read_csv("data.csv")
X = data.drop('target', axis=1)
y = data['target']

# Cross-validated accuracy
model = RandomForestClassifier(random_state=0)
scores = cross_val_score(model, X, y, cv=5)
print(f"CV accuracy: {scores.mean():.2f} +/- {scores.std():.2f}")

# Fit on all data and save the model
model.fit(X, y)
joblib.dump(model, 'rf.joblib')

# Later, load the model for inference
loaded = joblib.load('rf.joblib')
preds = loaded.predict(X.head())
print(preds)
```

Supervised Learning with k-nn

```
import pandas as pd
from sklearn.model_selection import cross_val_score
```

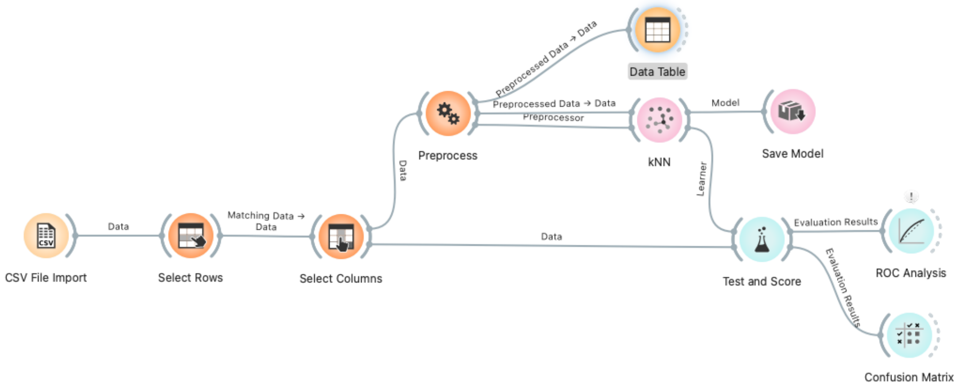

Figure 3: kNN workflow in Orange

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
import joblib

# Load data with a column named 'target'
data = pd.read_csv("data.csv")
X = data.drop('target', axis=1)
y = data['target']

# Scale features and evaluate k-NN via 5-fold cross-validation
model = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsClassifier(n_neighbors=5))
])
scores = cross_val_score(model, X, y, cv=5)
print(f"CV accuracy: {scores.mean():.2f} +/- {scores.std():.2f}")

# Fit on all data and save the model
model.fit(X, y)
joblib.dump(model, 'knn.joblib')

# Later, load the model for inference
loaded = joblib.load('knn.joblib')
preds = loaded.predict(X.head())
print(preds)
```