1.Supervised Learning

A. Regression

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()

from sklearn.linear_model import Ridge
model = Ridge()

from sklearn.linear_model import Lasso
model = Lasso()

from sklearn.linear_model import ElasticNet
model = ElasticNet()

from sklearn.svm import SVR
model = SVR()

from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()

from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()

from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor()

from xgboost import XGBRegressor
model = XGBRegressor()

from lightgbm import LGBMRegressor
model = LGBMRegressor()

from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor()

from sklearn.neural_network import MLPRegressor
model = MLPRegressor()

# For regression models
model.fit(X_train, y_train)
```

B. Classification

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()

from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier()

from sklearn.svm import SVC
model = SVC()

from sklearn.naive_bayes import GaussianNB
model = GaussianNB()

from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()

from xgboost import XGBClassifier
model = XGBClassifier()

from lightgbm import LGBMClassifier
model = LGBMClassifier()

from sklearn.neural_network import MLPClassifier
model = MLPClassifier()

# For classification models
model.fit(X_train, y_train)
```

2.Unsupervised Learning

A. Clustering

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)

from sklearn.cluster import DBSCAN
model = DBSCAN()

from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=3)

from sklearn.cluster import MeanShift
model = MeanShift()

from sklearn.mixture import GaussianMixture
model = GaussianMixture(n_components=3)

from sklearn.cluster import Birch
model = Birch()

# For clustering models (no y needed)
model.fit(X)
```

C. Dimensionality Reduction

```
from sklearn.decomposition import PCA
model = PCA(n_components=2)

from sklearn.decomposition import TruncatedSVD
model = TruncatedSVD(n_components=2)

from sklearn.decomposition import NMF
model = NMF(n_components=2)

from sklearn.manifold import TSNE
model = TSNE(n_components=2)

import umap
model = umap.UMAP(n_components=2)

from sklearn.decomposition import LatentDirichletAllocation
model = LatentDirichletAllocation(n_components=10)

# For dimensionality reduction models (no y needed)
model.fit(X)
```

3.Model Evaluvation/Selection

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5)

from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(model, param_grid, cv=5)

from sklearn.model_selection import RandomizedSearchCV
random_search = RandomizedSearchCV(model, param_distributions, n_iter=10,
cv=5)

from sklearn.metrics import classification_report
report = classification_report(y_true, y_pred)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)

from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_true, y_scores)

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_true, y_pred)

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

4.Deep Learning

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Conv2D, Flatten
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(input_dim,)))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32)

# For Keras models
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

5.Other Tools

```
from sklearn.feature_selection import SelectKBest
selector = SelectKBest(k=10)
X_selected = selector.fit_transform(X, y)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)

from sklearn.preprocessing import OneHotEncoder
onehot = OneHotEncoder()
X_encoded = onehot.fit_transform(X)

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

from sklearn.pipeline import Pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LogisticRegression())
])
```