Generalized Linear Models:

Linear Regression:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
```

Ridge Regression:

```
from sklearn.linear_model import Ridge

model = Ridge(alpha=1.0)
```

Lasso Regression:

```
from sklearn.linear_model import Lasso

model = Lasso(alpha=1.0)
```

ElasticNet:

```
from sklearn.linear_model import ElasticNet

model = ElasticNet(alpha=1.0, l1_ratio=0.5)
```

Support Vector Machines (SVM):

SVM Classifier:

```
from sklearn.svm import SVC

model = SVC(kernel='rbf')
```

SVM Regressor:

```
from sklearn.svm import SVR

model = SVR(kernel='linear')
```

Nearest Neighbors:

k-Nearest Neighbors (k-NN):

```
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor

knn_classifier = KNeighborsClassifier(n_neighbors=5)

knn_regressor = KNeighborsRegressor(n_neighbors=5)
```

Decision Trees:

Decision Tree Classifier:

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
```

Decision Tree Regressor:

```
from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor()
```

Ensemble Methods:

Random Forests:

```
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

rf_classifier = RandomForestClassifier(n_estimators=100)

rf_regressor = RandomForestRegressor(n_estimators=100)
```

Gradient Boosting:

```
from sklearn.ensemble import GradientBoostingClassifier,GradientBoostingRegressor

gb_classifier = GradientBoostingClassifier(n_estimators=100)

gb_regressor = GradientBoostingRegressor(n_estimators=100)
```

AdaBoost:

from sklearn.ensemble import AdaBoostClassifier, AdaBoostRegressor

adb_classifier = AdaBoostClassifier(n_estimators=100)

adb_regressor = AdaBoostRegressor(n_estimators=100)

Extra Trees Classifier/Regressor:

from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor

et_classifier = ExtraTreesClassifier(n_estimators=100)

et_regressor = ExtraTreesRegressor(n_estimators=100)

Naive Bayes:

Gaussian Naive Bayes:

from sklearn.naive_bayes import GaussianNB

model = GaussianNB()

Multinomial Naive Bayes:

from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()

Bernoulli Naive Bayes:

from sklearn.naive_bayes import BernoulliNB

model = BernoulliNB()

Neural Network Models:

Multi-layer Perceptron (MLP) Classifier/Regressor:

from sklearn.neural_network import MLPClassifier, MLPRegressor

mlp_classifier = MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000)

mlp_regressor = MLPRegressor(hidden_layer_sizes=(100,), max_iter=1000)

Unsupervised Learning Models:

Clustering:

```
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering, MeanShift

kmeans = KMeans(n_clusters=3)

dbscan = DBSCAN(eps=0.5, min_samples=5)

agg_clustering = AgglomerativeClustering(n_clusters=3)

meanshift = MeanShift()
```

Dimensionality Reduction:

```
from sklearn.decomposition import PCA, TruncatedSVD, FastICA

pca = PCA(n_components=2)

tsvd = TruncatedSVD(n_components=2)

ica = FastICA(n_components=2)
```

Model Selection and Evaluation:

Cross-validation techniques:

```
from sklearn.model_selection import cross_val_score, GridSearchCV,
RandomizedSearchCV

scores = cross_val_score(model, X, y, cv=5)

grid_search = GridSearchCV(estimator=model, param_grid={}, cv=5)

random_search = RandomizedSearchCV(estimator=model, param_distributions={},
cv=5)
```

Model evaluation metrics:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
```

precision = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred)

Preprocessing and Utilities:

Feature preprocessing:

from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotEncoder, LabelEncoder

scaler = StandardScaler()

minmax_scaler = MinMaxScaler()

onehot_encoder = OneHotEncoder()

label_encoder = LabelEncoder()

Imputation:

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')

Model Pipelines:

from sklearn.pipeline import Pipeline, FeatureUnion

pipeline = Pipeline(steps=[('scaler', StandardScaler()), ('model', model)])

feature_union = FeatureUnion([('pca', PCA()), ('tsvd', TruncatedSVD())])