Training and Test Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=0)
```

Data Preprocessing

Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
standardized_X_train = scaler.transform(X_train)
standardized_X_test = scaler.transform(X_test)
```

Normalization

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

Imputing Missing Values

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

Dummy Encoding

```
import pandas as pd
dummy_var = pd.get_dummies(data, drop_first = True)
```

Model Fitting

```
model.fit(X_train, y_train)
```

Prediction

```
y_pred = model.predict(X_test)
```

Stacking Classifier

```
from sklearn.ensemble import StackingClassifier
clf = StackingClassifier(estimators=base_learners, final_estimator)
```

Classification Models

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression()
import statsmodels.api as sm
logistic_model = sm.Logit(y_train, X_train).fit()
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt_clf = DecisionTreeClassifier()
from sklearn.tree import DecisionTreeRegressor
dt_regr = DecisionTreeRegressor()
```

Random Forest

```
from sklearn.tree import DecisionTreeClassifier
dt_clf = DecisionTreeClassifier()
from sklearn.ensemble import RandomForestRegressor
rf _regr = RandomForestRegressor
```

K Nearest Neighbors

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

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
from sklearn.naive_bayes import BernoulliNB
bnb = BernoulliNB()
```

AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier
ada_clf = AdaBoostClassifier(model, n_estimators)
```

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
gboost = GradientBoostingClassifier(n_estimators, max_depth)
```

Extreme Gradient Boosting

```
from xgboost import XGBClassifier
xgb_model = XGBClassifier()
```

Confusion Matrix

	Negative	Positive
Negative	True Negative (TN)	False Positive (FP) Type 1 Error
Positive	False Negative (FN) Type 2 Error	True Positive (TP)

Sensitivity (Recall) = TP / (TP+FN)

Precision = TP / (TP+FN)

Accuracy = (TP+TN) / (TP+TN+FN+FP)

Specificity = TN / (TN+FP)

F1-Score = 2(Precision * Recall) / (Precision + Recall)

Evaluating Model Performance

Regression Metrics

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_true, y_pred)
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
from sklearn.metrics import r2_score
r2_score(y_true, y_pred)
```

Evaluating Model Performance

Classification Metrics

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
from sklearn.metrics import classification_report
classification_report(y_test, y_pred)
```