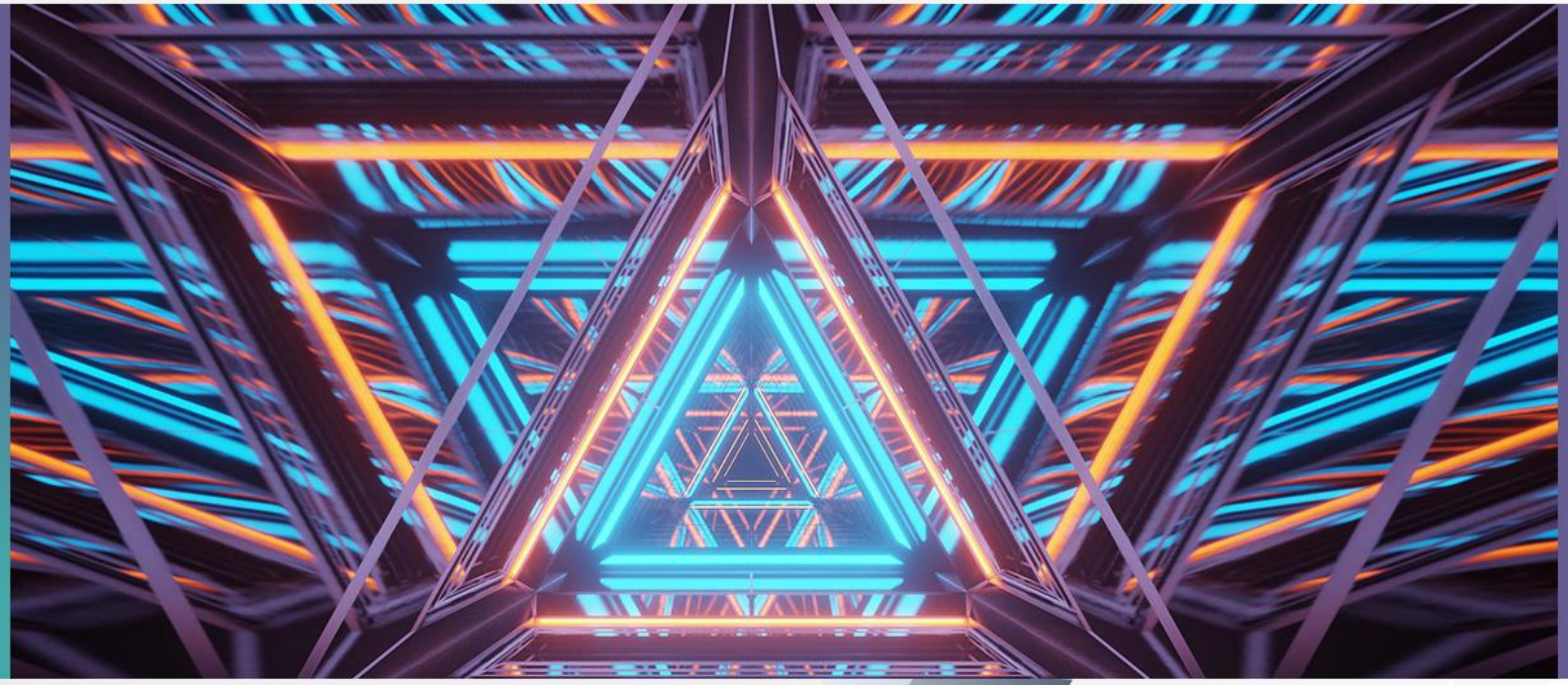
TAUTOLOGY
INNOVATION
SCHOOL

TAUTOLOGY

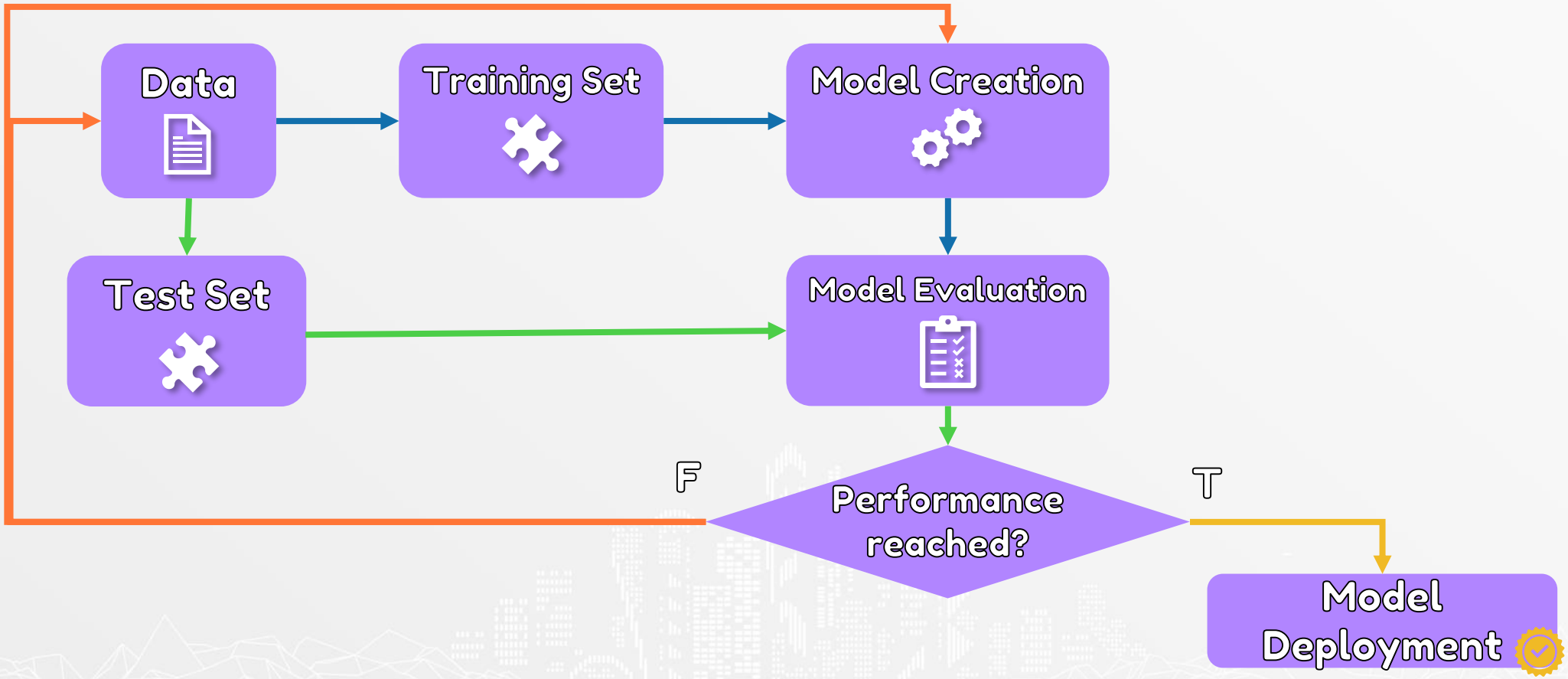
CODE PIPELINE



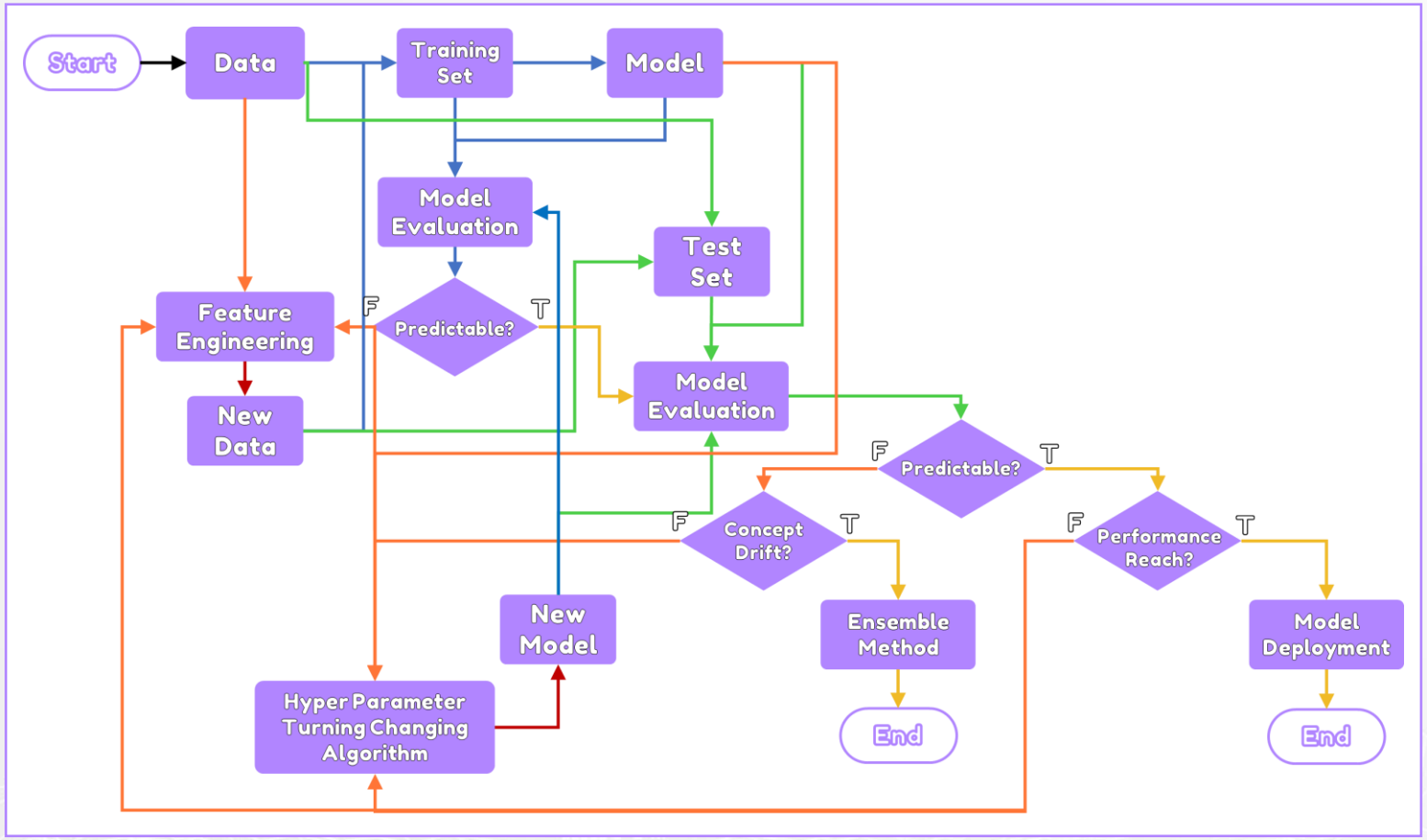
CODE PIPELINE

BY TAUTOLOGY

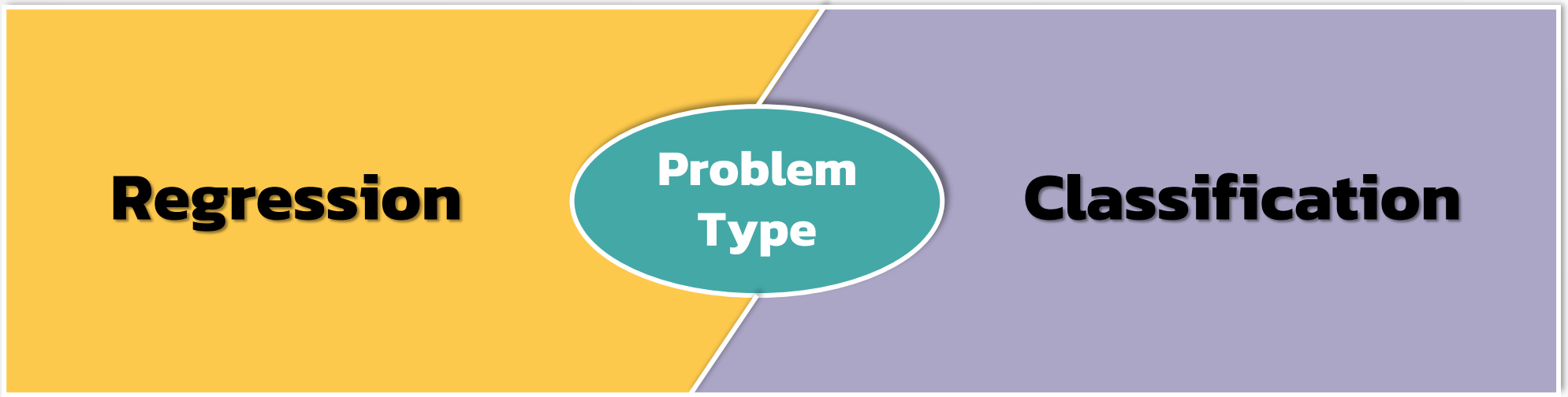
Supervised Learning Workflow



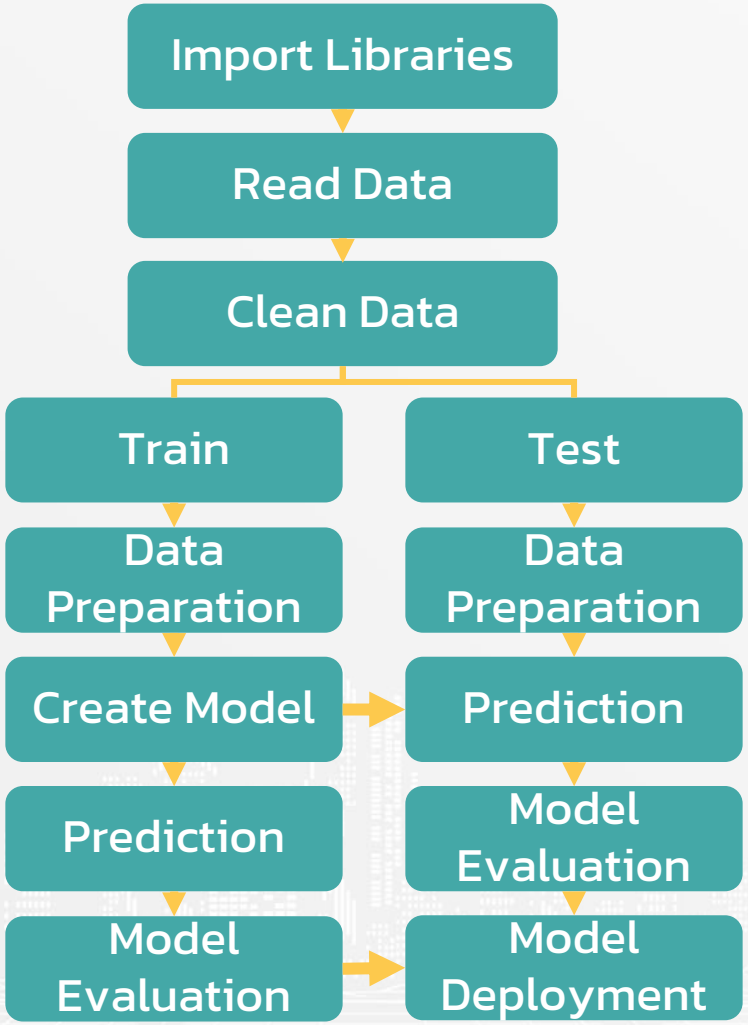
Supervised Learning Workflow

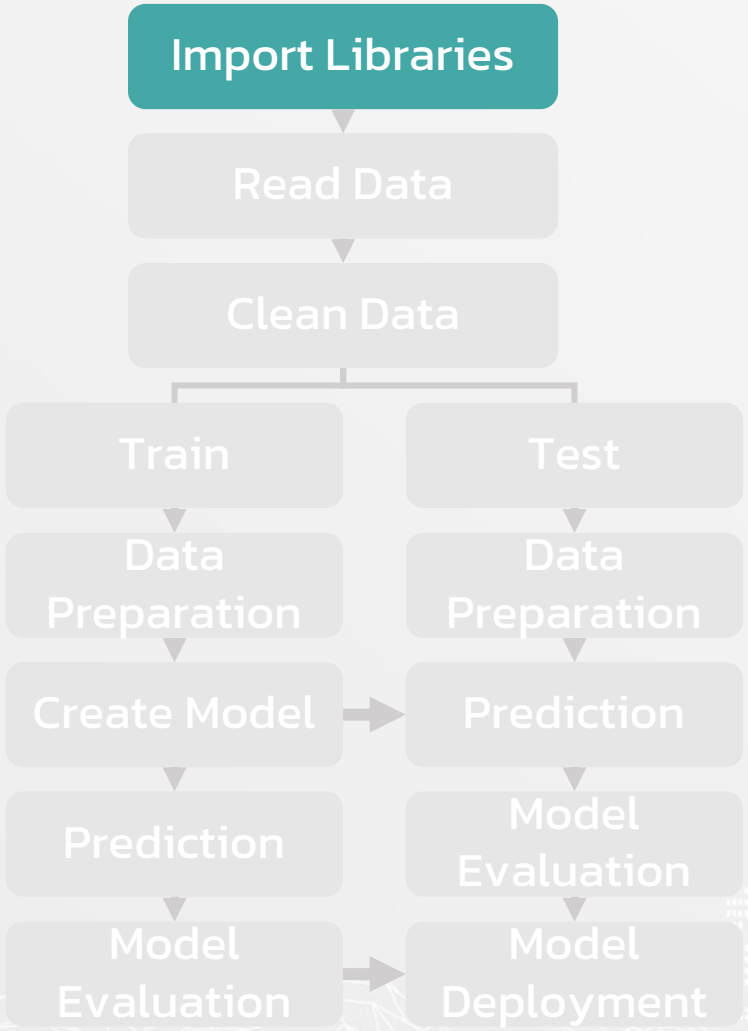


Workshop Overview



Code Pipeline





Import Libraries

Regression

Classification

1. NumPy
2. pandas
3. scikit learn
4. matplotlib

Code

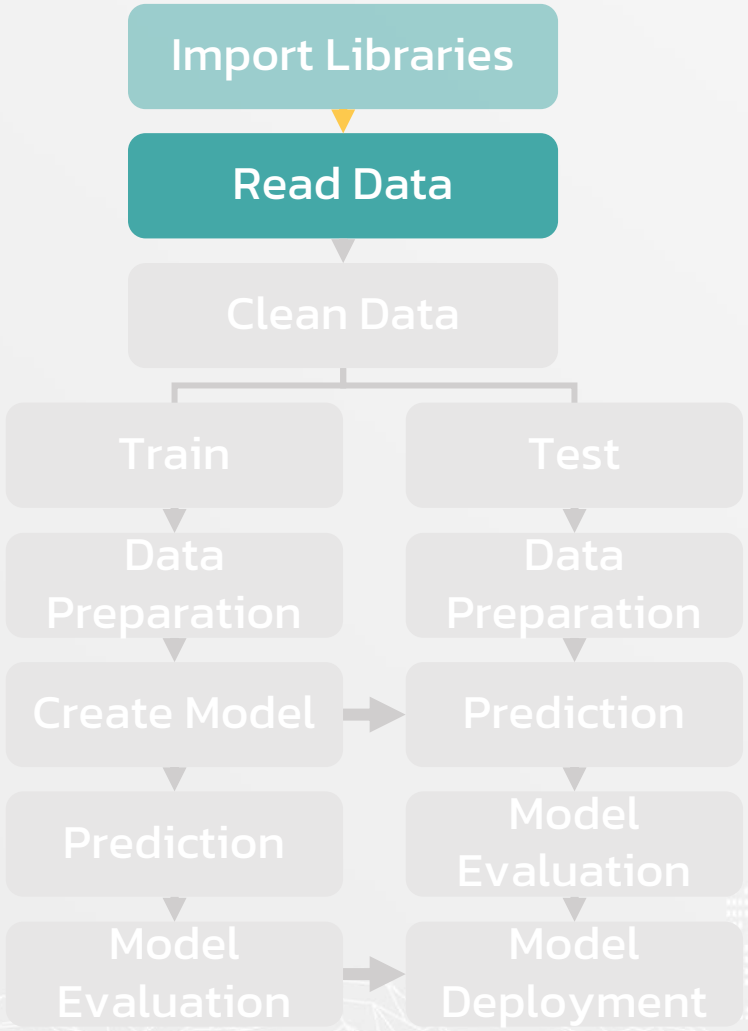
Regression

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, mean_absolute_percentage_error

import warnings
warnings.filterwarnings('ignore')

np.random.seed(12345)
```

Code

Classification

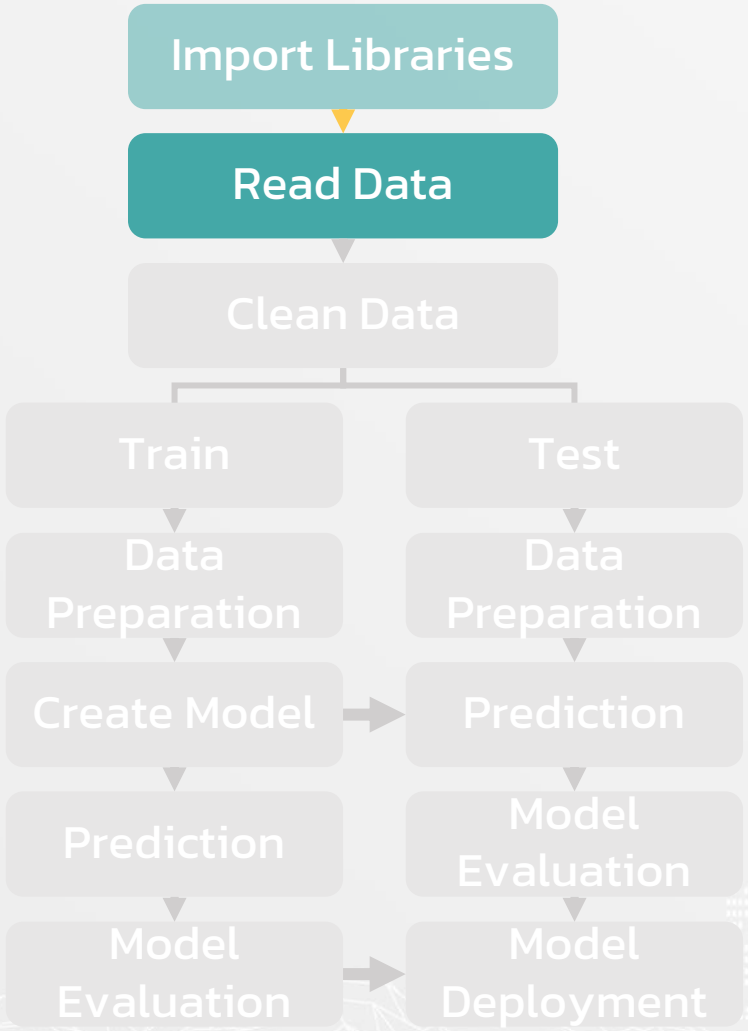
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, StandardScaler, MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.discriminant_analysis import LinearDiscriminant Analysis
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import plot_confusion_matrix, classification_report

import warnings
warnings.filterwarnings('ignore')

np.random.seed(12345)
```



Read Data

Regression

	age	experience	gpa	degree	position	salary
0	30.0	7.0	3.94	bachelor	engineer	32500.0
1	26.0	2.0	2.86	bachelor	NaN	22500.0
2	27.0	0.0	3.13	doctorate	secretary	37000.0
3	32.0	NaN	3.10	bachelor	engineer	24500.0
4	24.0	1.0	3.81	bachelor	accountant	23500.0
5	35.0	7.0	3.93	doctorate	secretary	43500.0
6	23.0	1.0	3.78	master	accountant	30500.0
7	32.0	8.0	3.04	bachelor	accountant	31500.0
8	27.0	2.0	3.52	bachelor	secretary	18500.0
	⋮	⋮	⋮	⋮	⋮	⋮



Read Data

Classification

	age	experience	gpa	degree	position	expected_salary	result
0	29.0	7	2.71	bachelor	secretary	19500	accept
1	29.0	4	NaN	bachelor	secretary	20500	accept
2	27.0	2	3.40	doctorate	accountant	43000	reject
3	33.0	11	3.25	bachelor	NaN	39000	reject
4	34.0	9	3.22	master	engineer	46500	reject
5	26.0	0	3.99	bachelor	accountant	29500	reject
6	23.0	1	3.60	bachelor	accountant	22000	accept
7	27.0	3	2.64	doctorate	accountant	33000	accept
8	23.0	0	2.69	bachelor	secretary	21500	reject
9	25.0	0	2.88	bachelor	engineer	29500	reject
	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Code

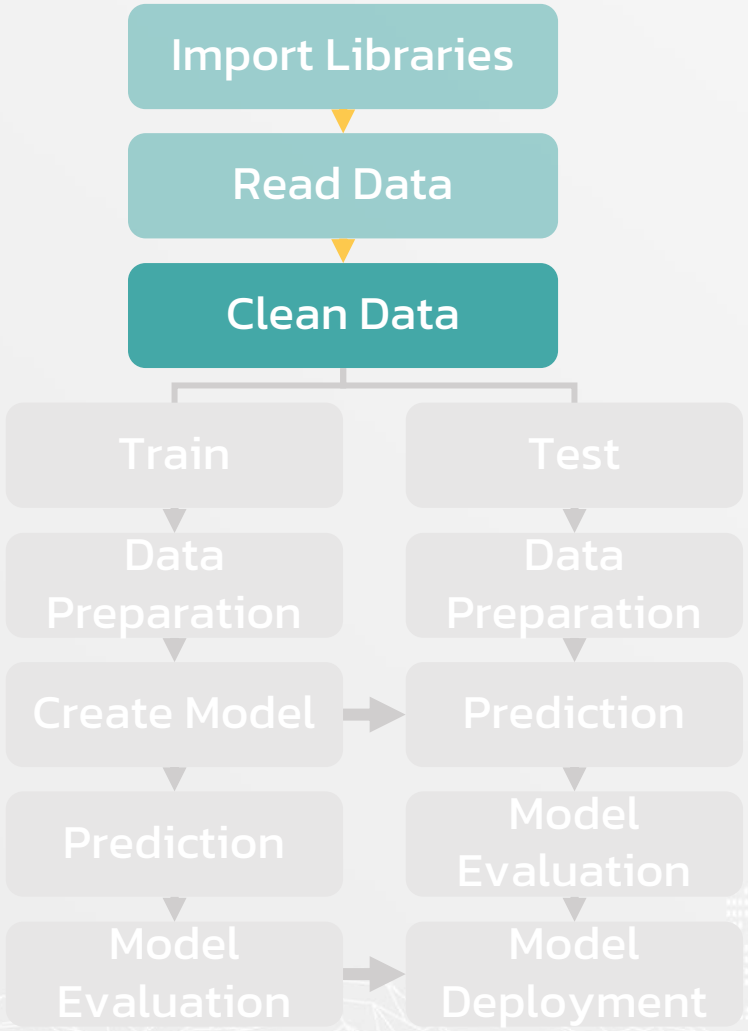
Regression

```
1  data = pd.read_csv('salary_dataset.csv')
```

Code

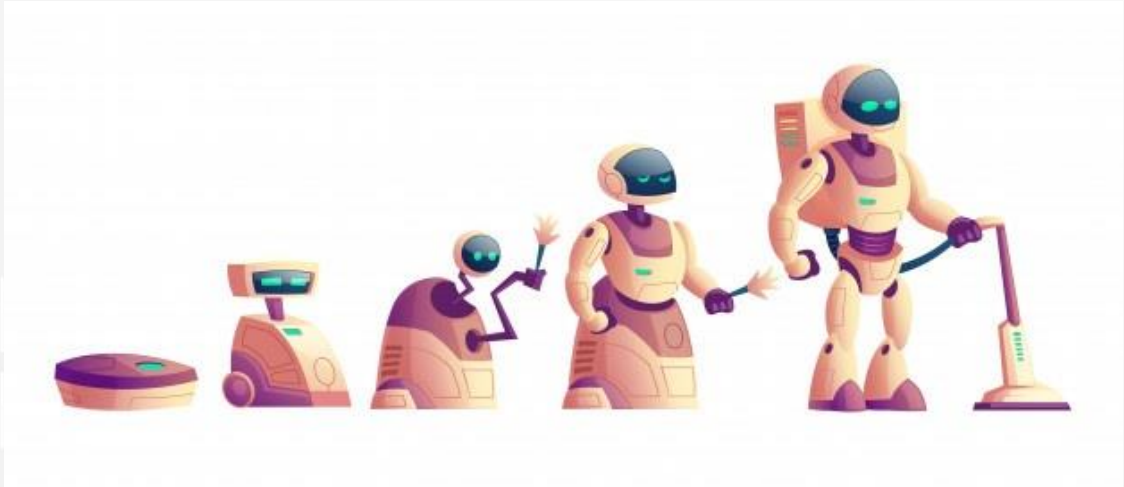
Classification

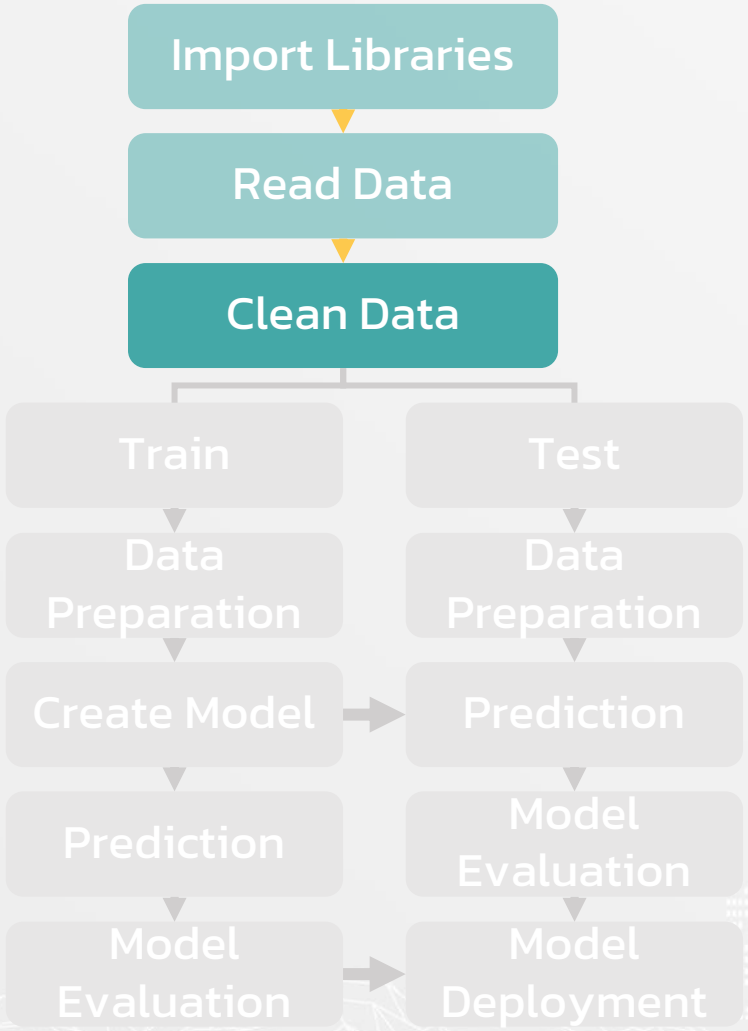
```
1  data = pd.read_csv('job_acceptance_dataset.csv')
```



Clean Data

1. Handle Missing Values
2. Handle Outliers





Clean Data

1. **Handle Missing Values**
2. Handle Outliers

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         89 non-null     float64
 1   experience  89 non-null     float64
 2   gpa         90 non-null     float64
 3   degree      90 non-null     object
 4   position    89 non-null     object
 5   salary      89 non-null     float64
dtypes: float64(4), object(2)
memory usage: 4.3+ KB
```

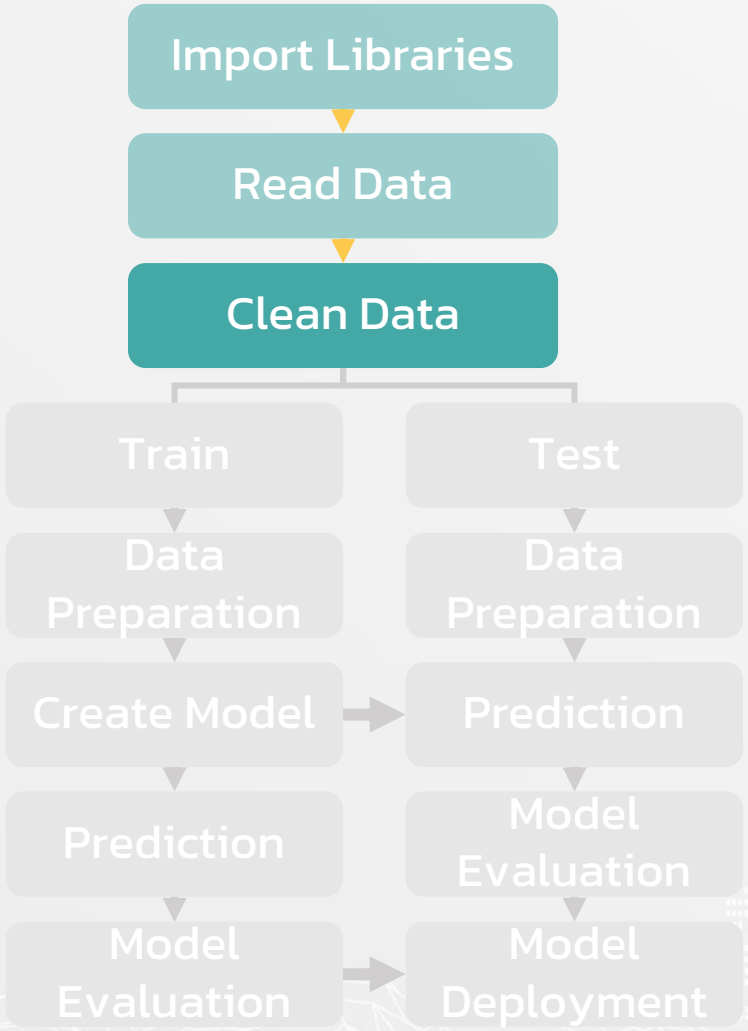
Code

- Check Missing Values

```
1  data.info()
```

- Remove Missing Values

```
1  data.dropna(axis=0, inplace=True)
```



Clean Data

1. Handle Missing Values
2. **Handle Outliers**

	age	experience	gpa	salary
count	86.000000	86.000000	86.000000	86.000000
mean	28.023256	3.848837	3.278605	31348.837209
std	4.408486	3.702201	0.528937	9255.227384
min	21.000000	0.000000	2.540000	13000.000000
25%	24.000000	1.000000	2.820000	24500.000000
50%	28.000000	3.000000	3.260000	30500.000000
75%	32.000000	7.000000	3.640000	37375.000000
max	35.000000	13.000000	5.880000	54000.000000

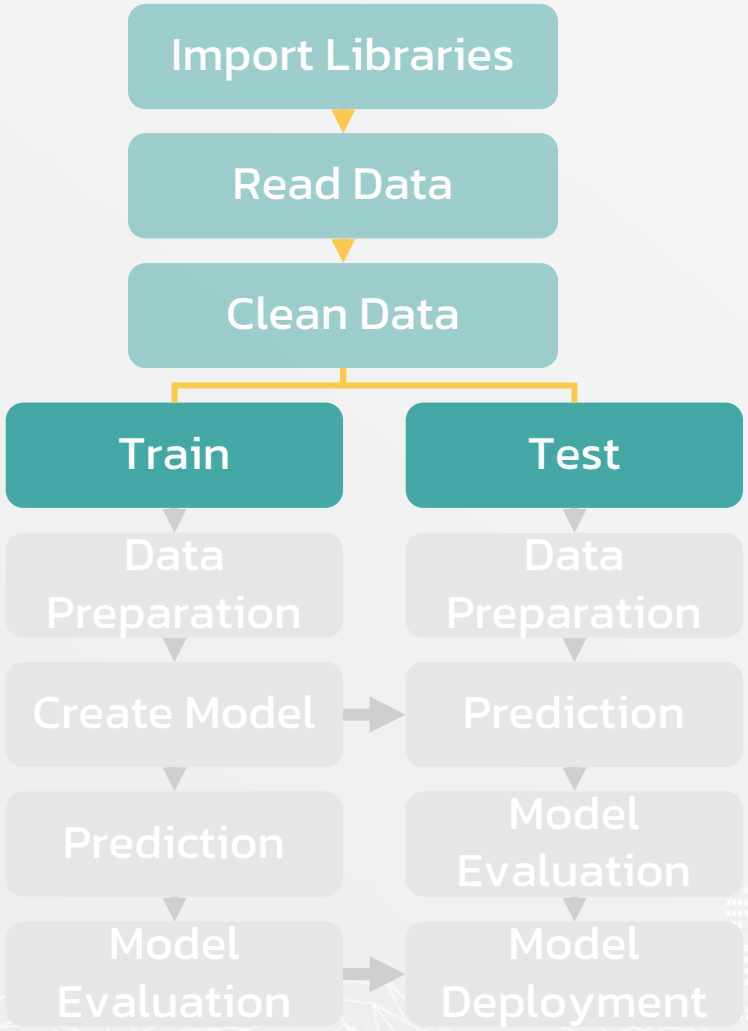
Code

- Check Outliers

```
1  data.describe()
```

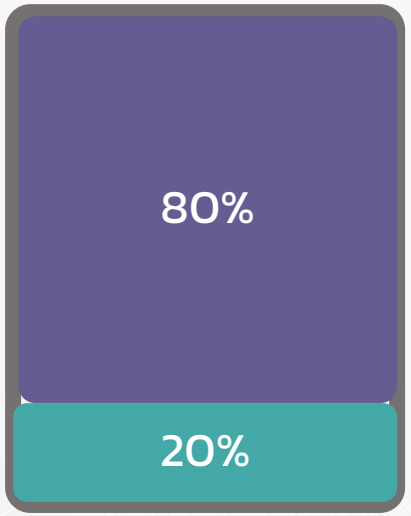
- Remove Outliers

```
1  _filter = data['gpa'] <= 4.00
2  data = data[_filter]
```



Train/Test

แบ่งข้อมูลออกเป็น 2 ชุด คือ training set และ test set ด้วยอัตราส่วน 80:20 ตามลำดับ

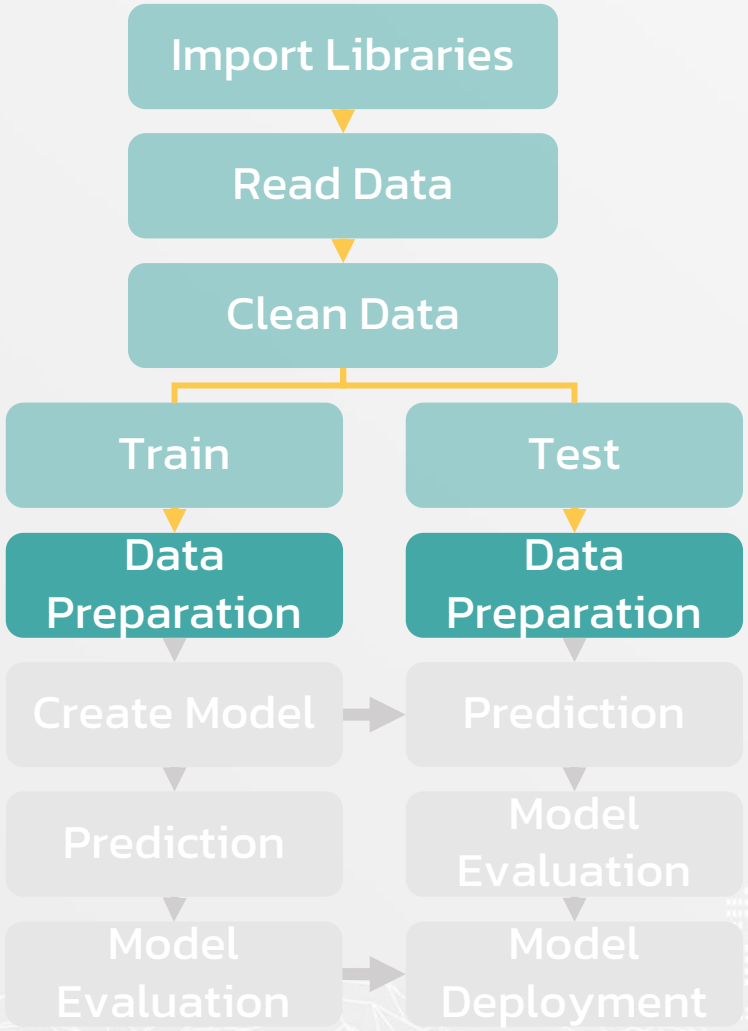


Code

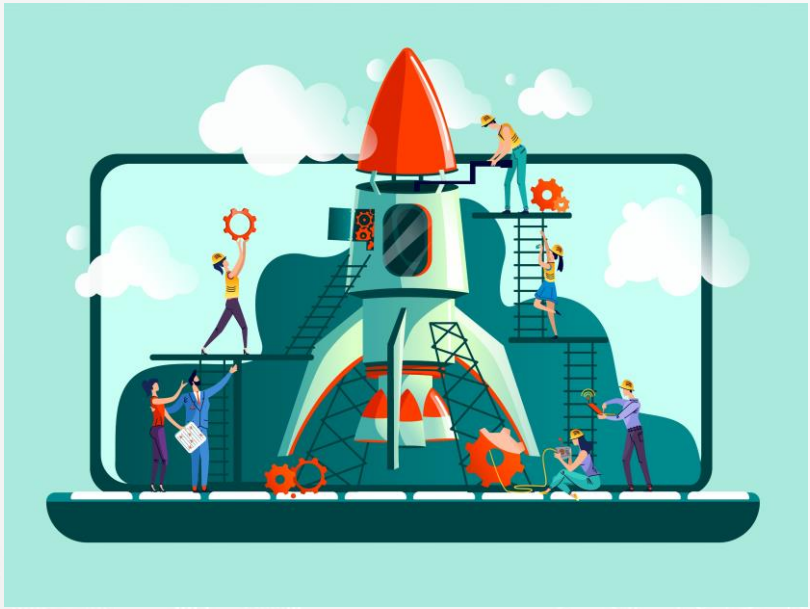
```python
target_name = 'salary'
feature_name = list(data.columns.drop(target_name))
```

```python
X = data[feature_name]
y = data[target_name]
```

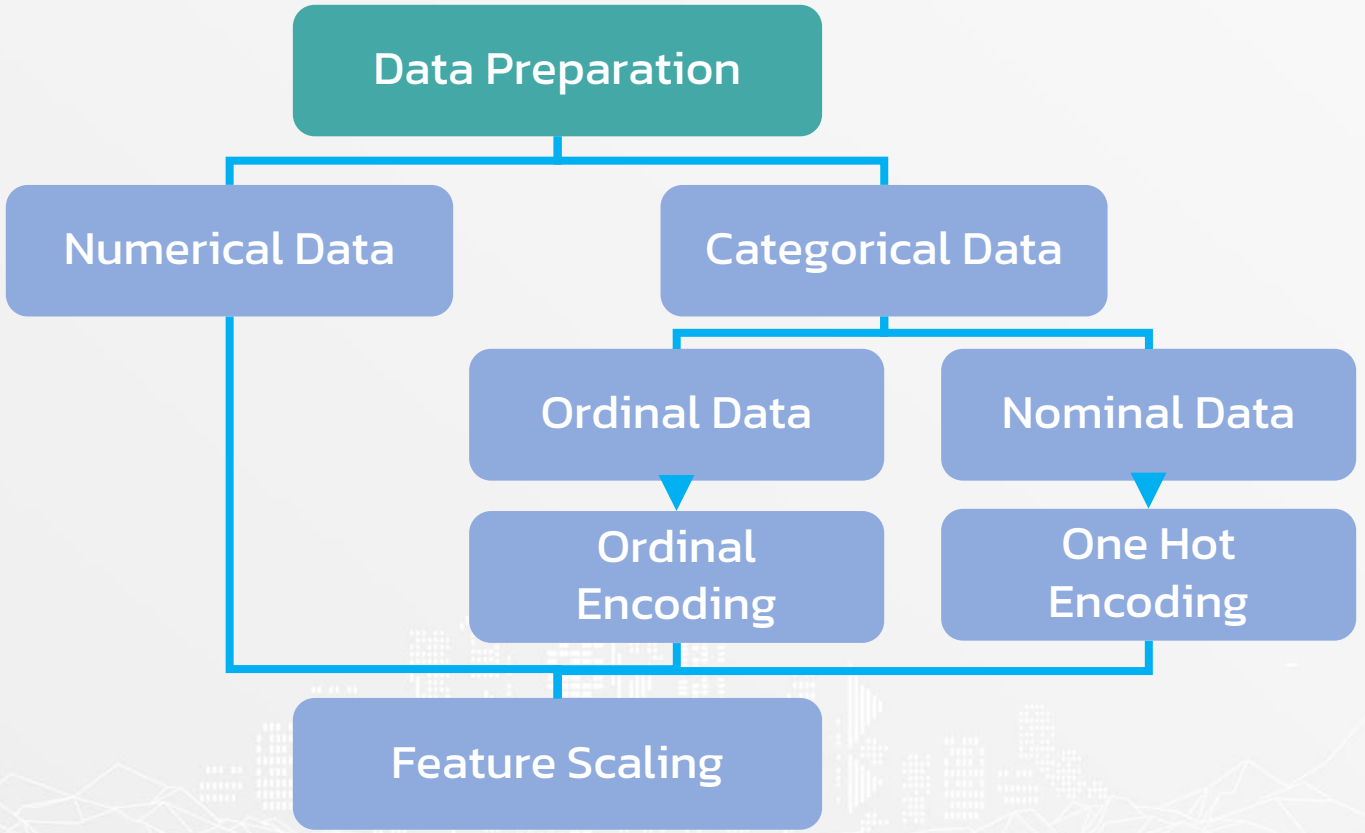
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, shuffle=True)
```



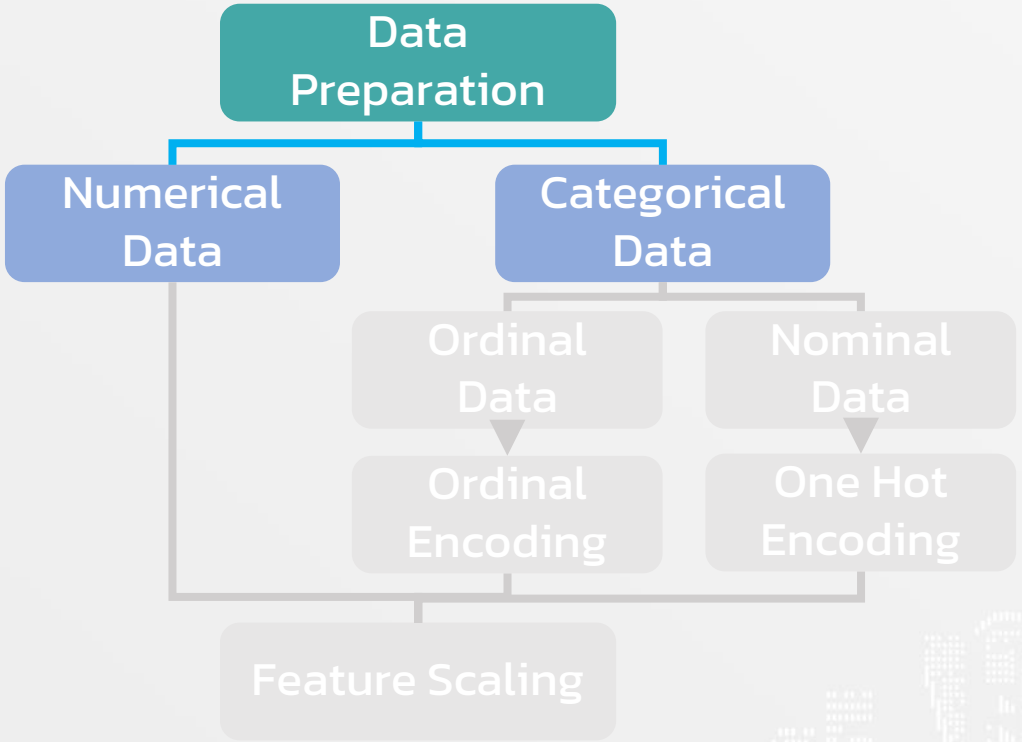
Data Preparation



Data Preparation

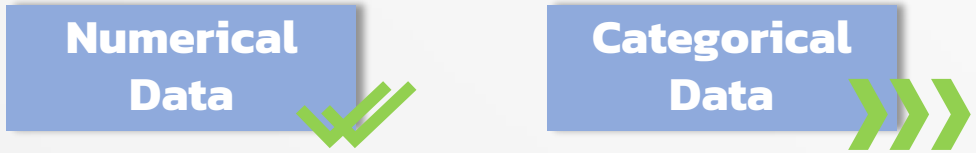


Data Preparation



Type of Features

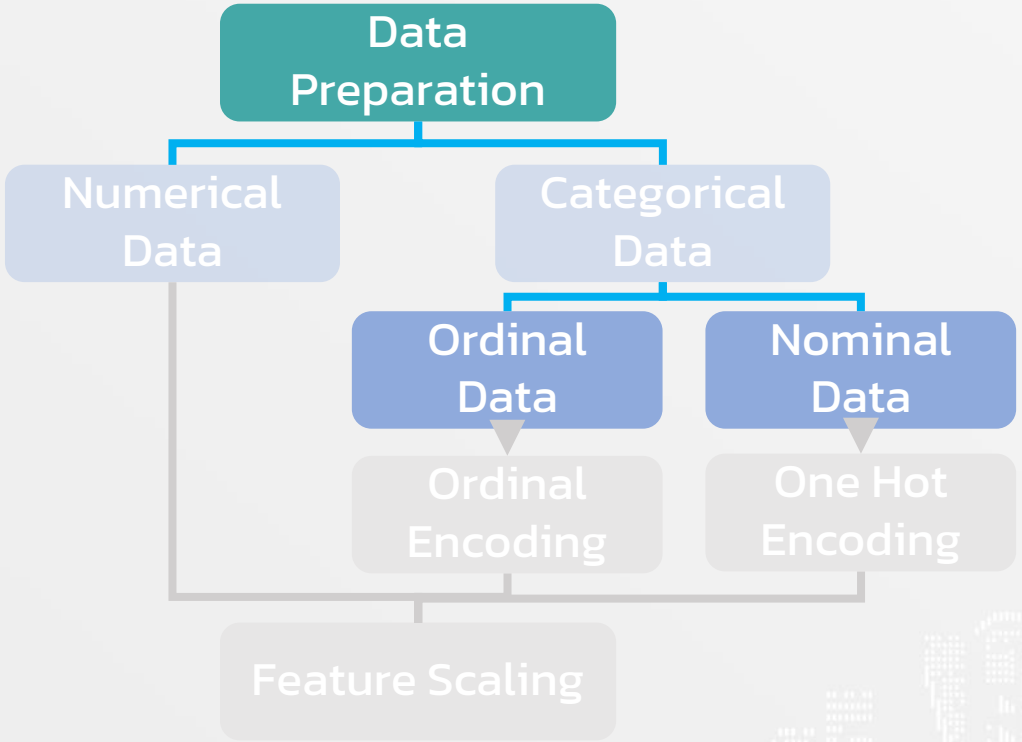
พิจารณาและจำแนก feature ที่มีลักษณะข้อมูลแบบ numerical data และ categorical data ออกจากกัน



Code

```
numerical_feature = ['age', 'experience', 'gpa']
categorical_feature = ['degree', 'position']
```

Data Preparation



Type of Categorical Features

พิจารณาและจำแนก feature ที่มีลักษณะข้อมูลแบบ ordinal data และ nominal data ออกจากกัน

```
degree : ['bachelor' 'doctorate' 'master']
position : ['accountant' 'engineer' 'secretary']
```

Code

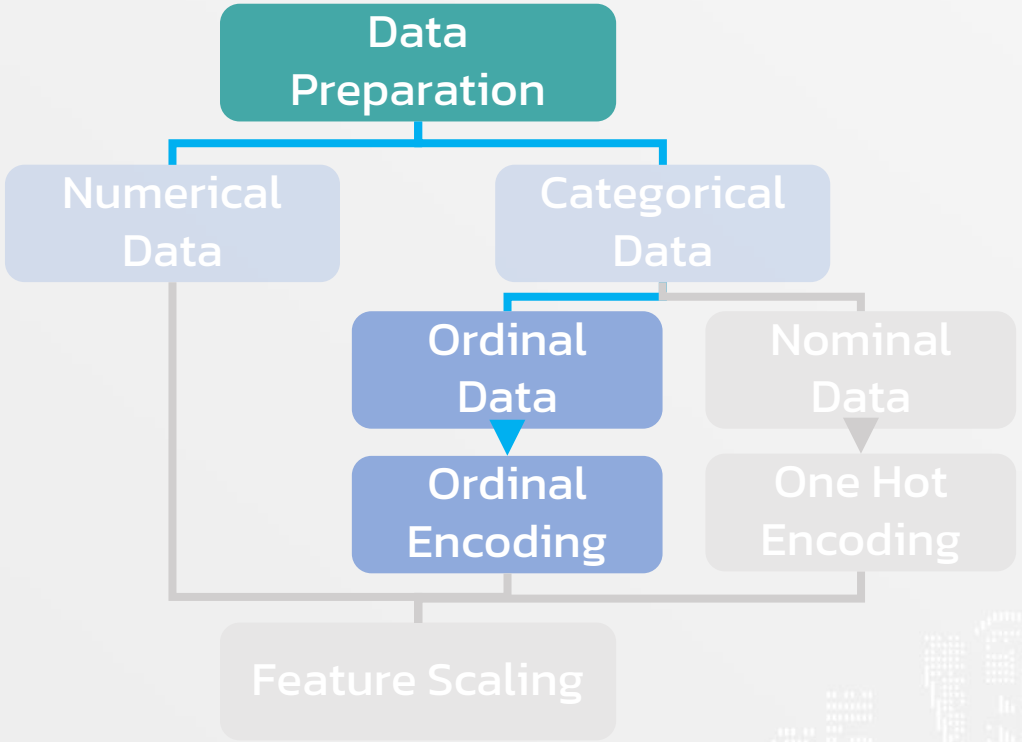
- Consider each Feature of Categorical Features

```
1 for feature in categorical_feature:
2     print(feature, ':', np.unique(X_train[feature]))
```

- Classify into Ordinal Feature and Nominal Feature

```
1 ordinal_feature = ['degree']
2 nominal_feature = ['position']
```

Data Preparation



Ordinal Encoding

การทำ ordinal encoding จะต้องทำแบบเดียวกัน ทั้งใน training set และ test set

	degree
0	bachelor
1	master
2	bachelor
3	bachelor
4	bachelor

	degree
0	0.0
1	1.0
2	0.0
3	0.0
4	0.0

Code

- Ordinal Data

```
1  ordinal_feature = ['degree']
```

- Ordinal Encoding

```
1  categories = [
2      np.array(['bachelor', 'master', 'doctorate'], dtype=object)
3  ]
```

Code

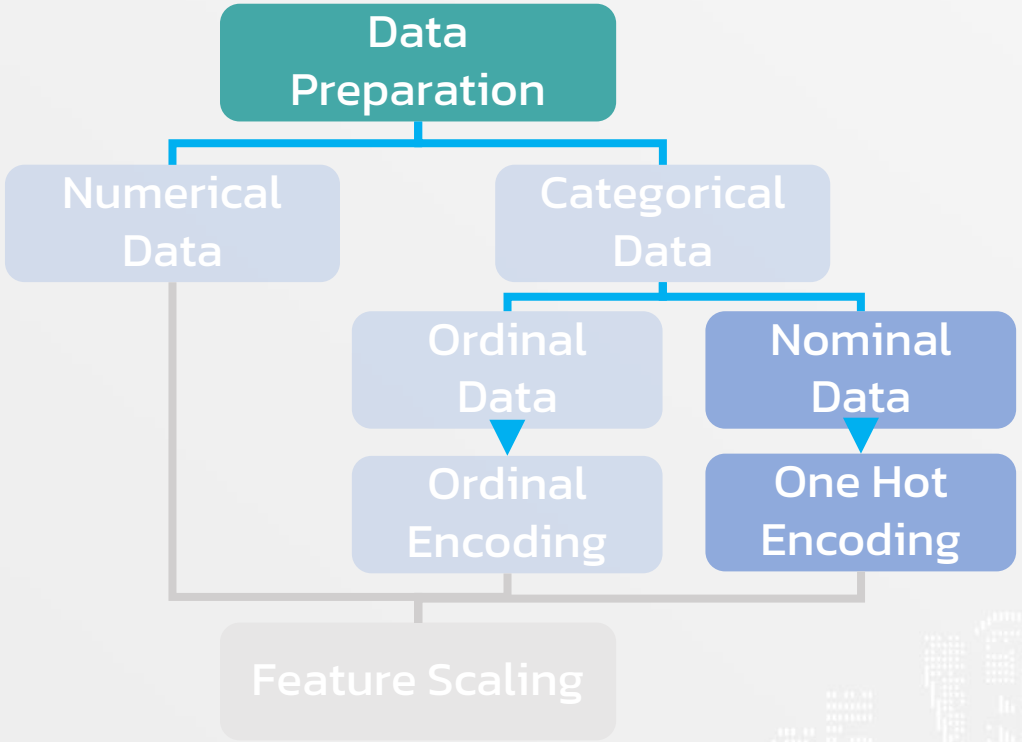
- Ordinal Encoding for **training set**

```
ordinal_encoder = OrdinalEncoder(categories=categories)
X_train[ordinal_feature] = ordinal_encoder.fit_transform(X_train[ordinal_feature])
```

- Ordinal Encoding for **test set**

```
X_test[ordinal_feature] = ordinal_encoder.transform(X_test[ordinal_feature])
```

Data Preparation



One Hot Encoding

การทำ one hot encoding จะต้องทำแบบเดียวกันทั้งใน training set และ test set

	position
0	secretary
1	secretary
2	engineer
3	engineer
4	secretary

	position_accountant	position_engineer	position_secretary
0	0.0	0.0	1.0
1	0.0	0.0	1.0
2	0.0	1.0	0.0
3	0.0	1.0	0.0
4	0.0	0.0	1.0

Code

- Nominal Data

```
1 nominal_feature = ['position']
```

- One Hot Encoding

```
1 one_hot_encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
2 one_hot_encoder.fit(X_train[nominal_feature])
```

```
1 one_hot_feature = []
2 for i, feature in enumerate(nominal_feature):
3     for cate in one_hot_encoder.categories_[i]:
4         one_hot_feature_name = str(feature) + '_' + str(cate)
5         one_hot_feature.append(one_hot_feature_name)
```

Code

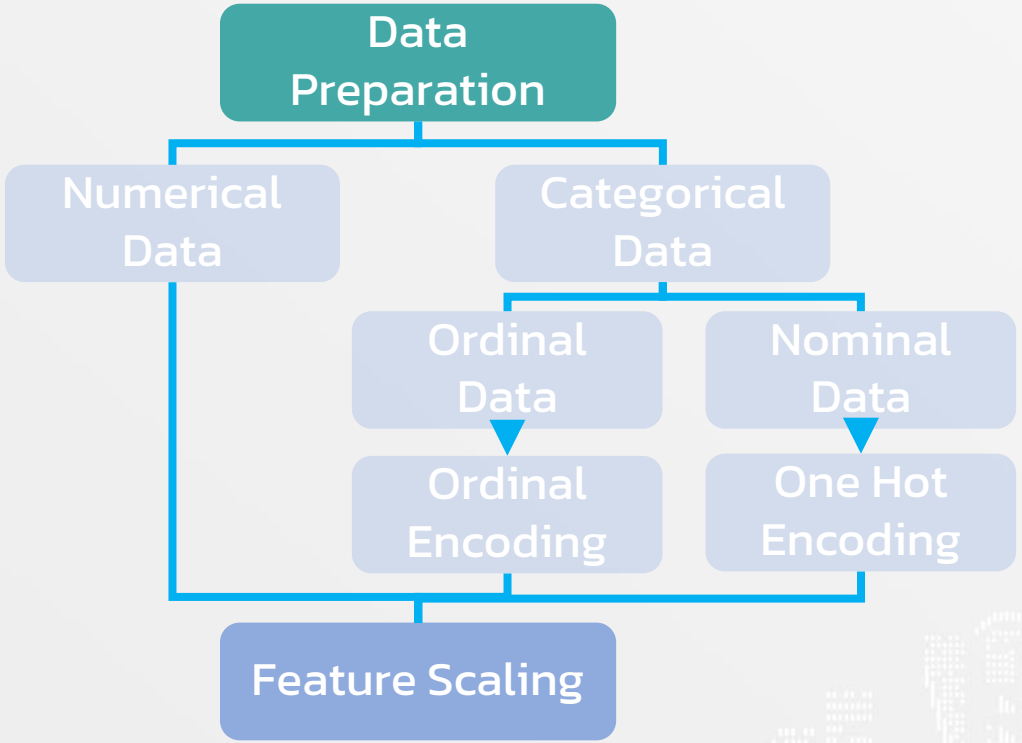
- One Hot Encoding for **training set**

```
X_train[one_hot_feature] = one_hot_encoder.transform(X_train[nominal_feature])
X_train.drop(nominal_feature, axis=1, inplace=True)
```

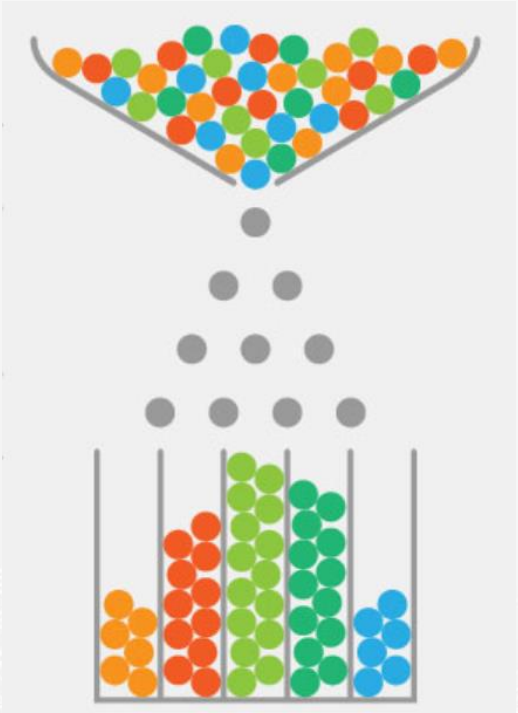
- One Hot Encoding for **test set**

```
X_test[one_hot_feature] = one_hot_encoder.transform(X_test[nominal_feature])
X_test.drop(nominal_feature, axis=1, inplace=True)
```

Data Preparation



Feature Scaling



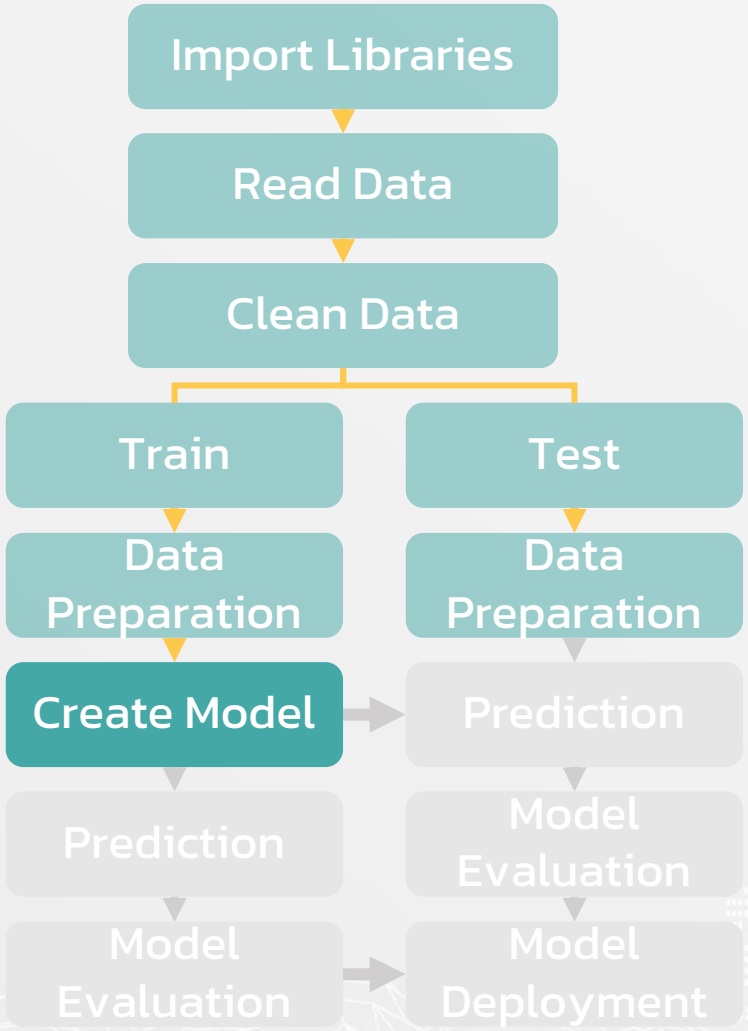
Code

- Feature Scaling for **training set**

```
1 scaler = StandardScaler()
2 X_train_scaled = scaler.fit_transform(X_train)
```

- Feature Scaling for **test set**

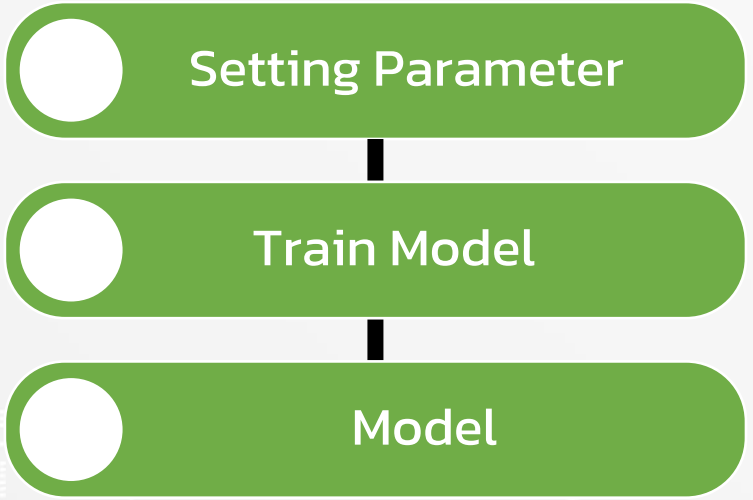
```
1 scaler = StandardScaler()
2 X_test_scaled = scaler.transform(X_test)
```



Create Model

Regression

Classification



Code – Setting Parameter

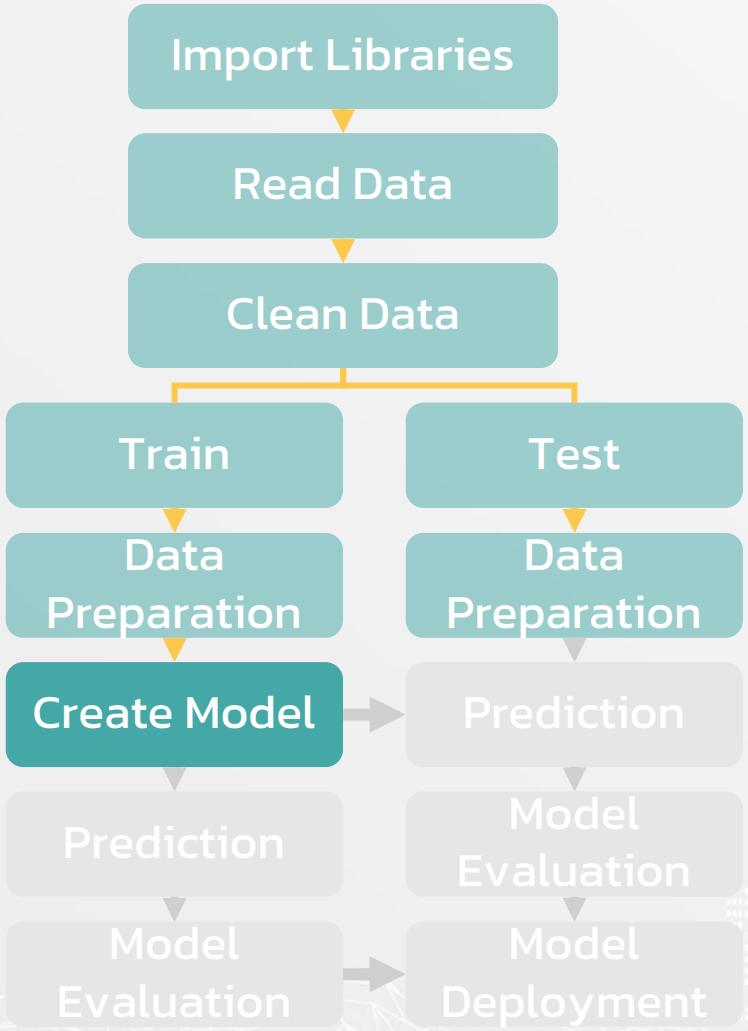
Regression

```
reg = LinearRegression()
reg = MLPRegressor()
reg = DecisionTreeRegressor()
reg = SVR()
reg = GaussianProcessRegressor()
```

Code – Setting Parameter

Classification

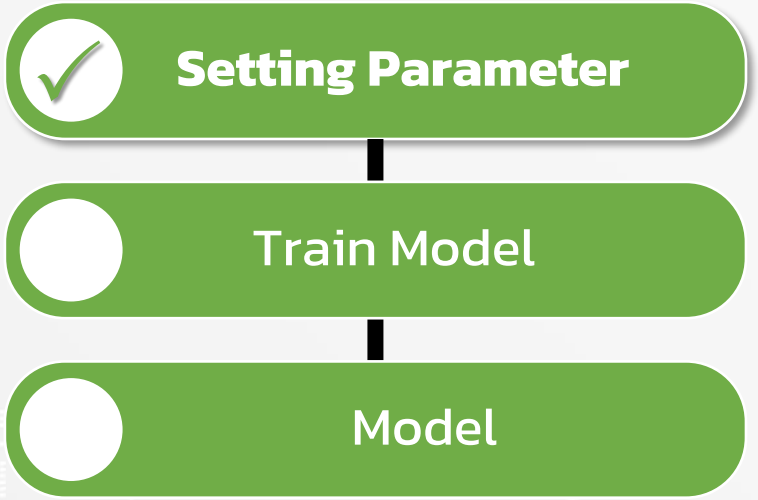
```
clf = LogisticRegression()
clf = MLPClassifier()
clf = DecisionTreeClassifier()
clf = KNeighborsClassifier()
clf = SVC()
clf = LinearDiscriminantAnalysis()
clf = GaussianNB()
```



Create Model

Regression

Classification



Code – Train Model

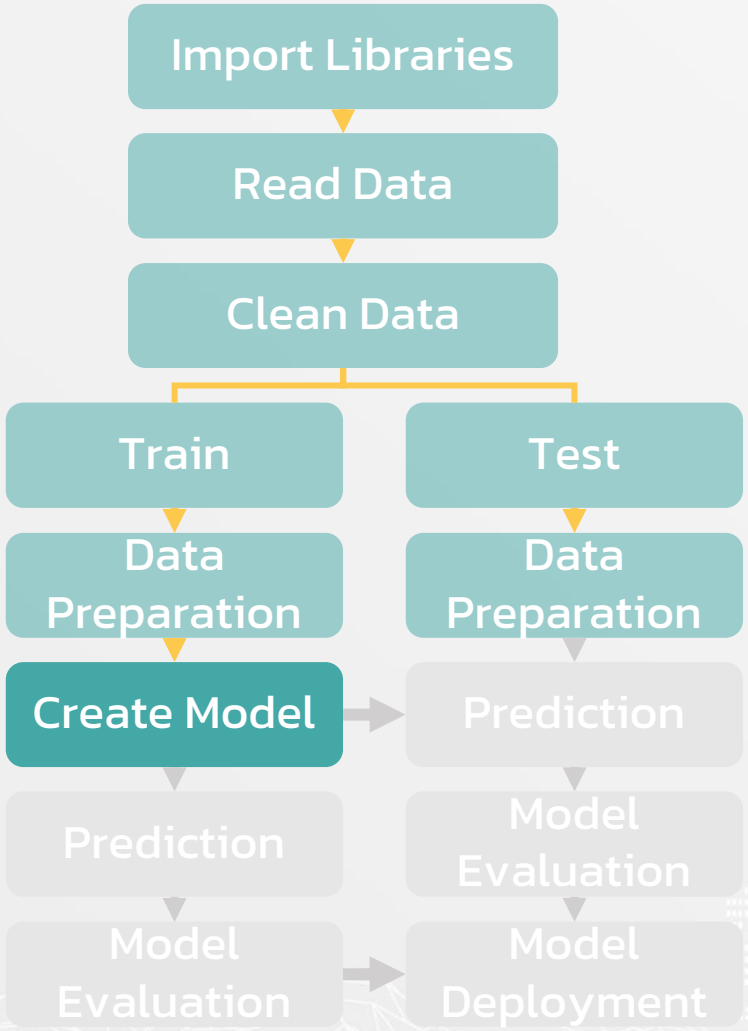
Regression

```
1  reg.fit(X_train, y_train)
```

Code – Train Model

Classification

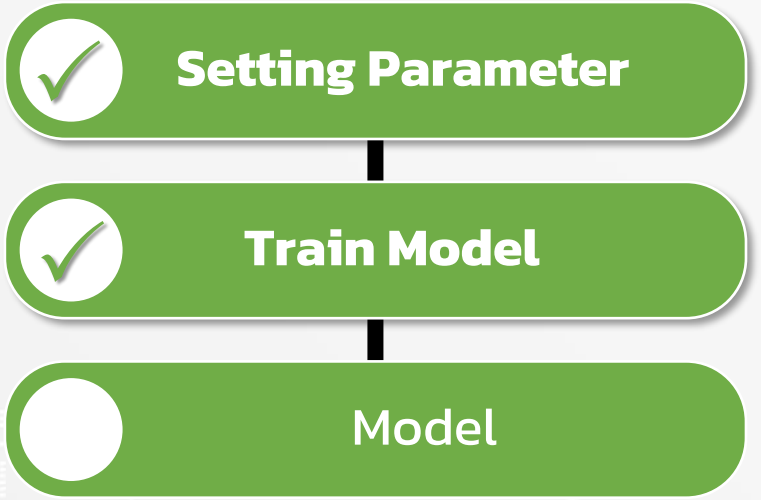
```
1  clf.fit(X_train_scaled, y_train)
```



Create Model

Regression

Classification



Code – Model

Regression

- Bias (w_0)

```
1 reg.intercept_
```

- Weight ($w_1, \dots, w_p$)

```
1 reg.coef_
```

Code – Model

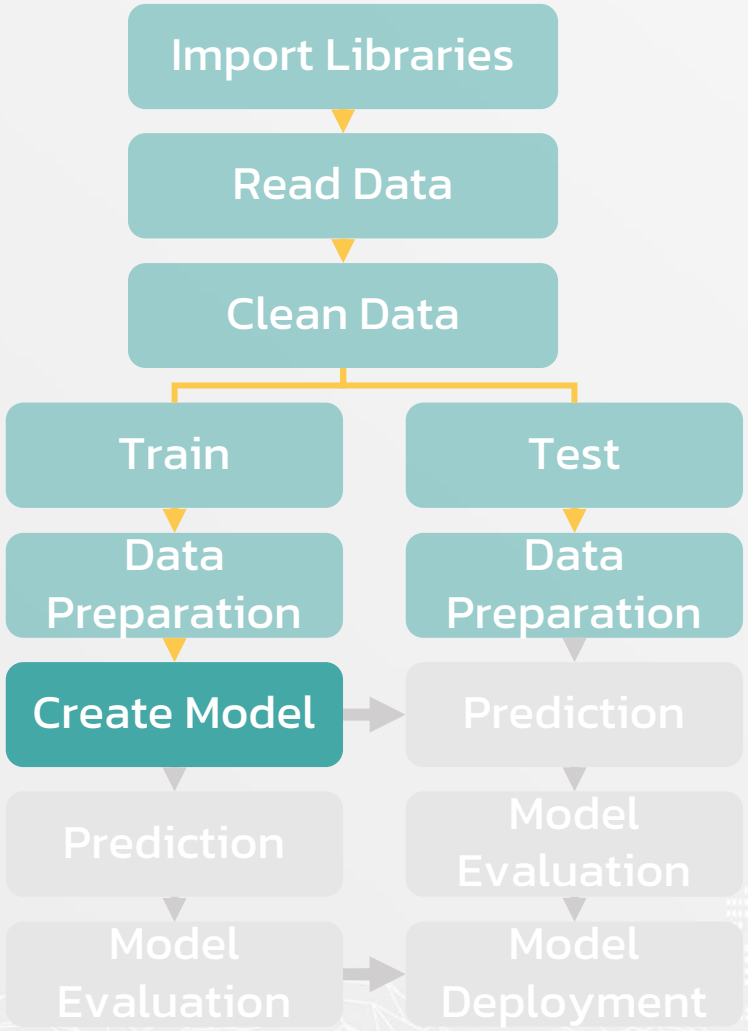
Classification

- Bias (w_0)

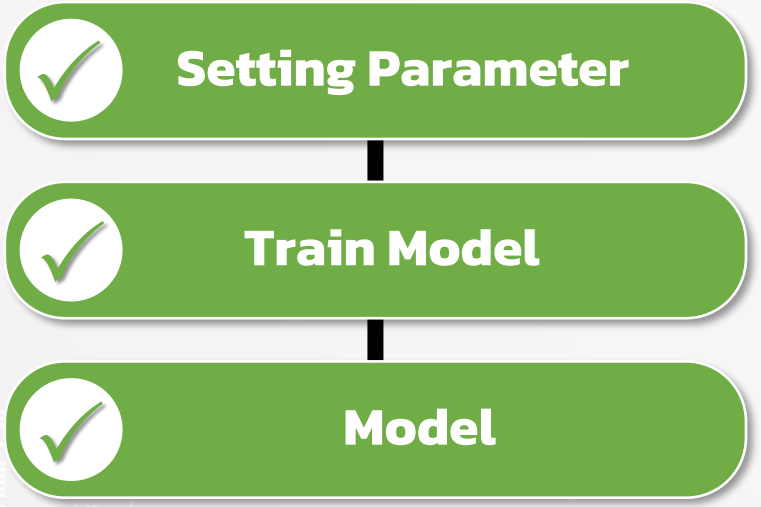
```
1 clf.intercept_
```

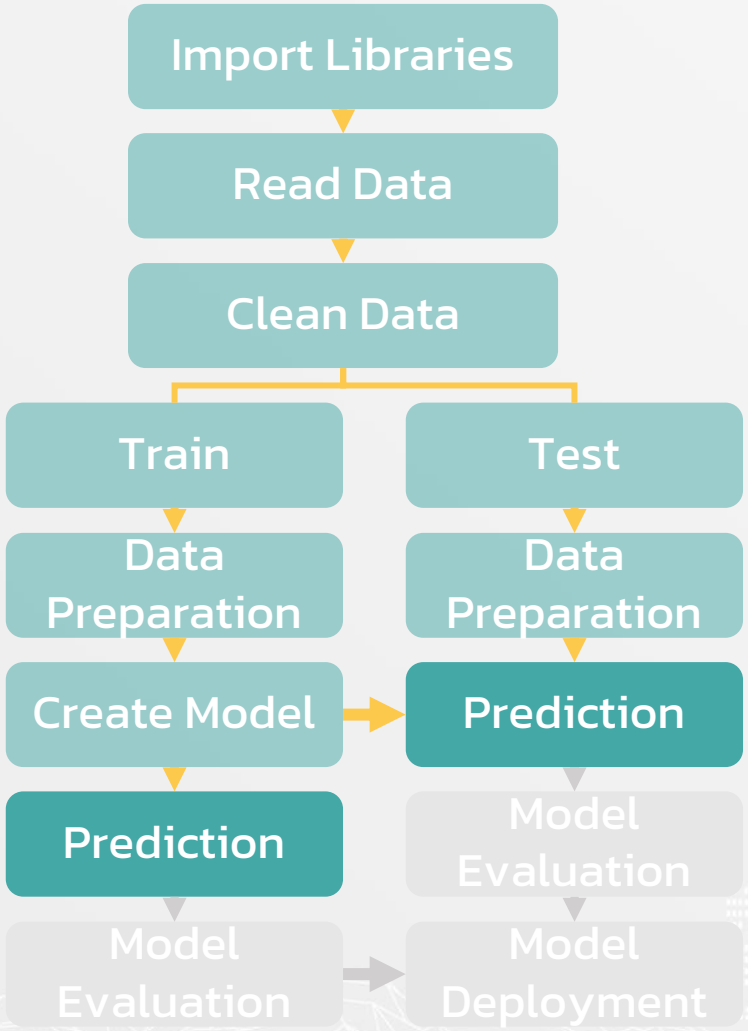
- Weight ($w_1, \dots, w_p$)

```
1 clf.coef_
```



Create Model





Prediction

Regression

Classification

Code

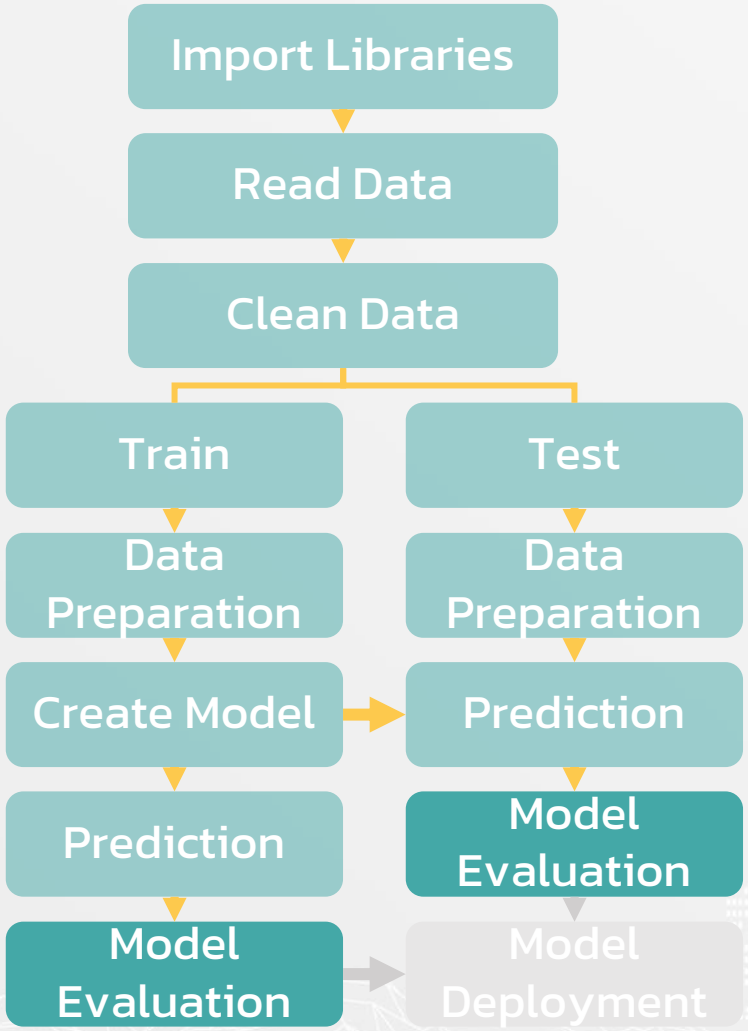
Regression

- Prediction for **training set**

```
1 y_pred_train = reg.predict(X_train)
```

- Prediction for **test set**

```
1 y_pred_test = reg.predict(X_test)
```



Model Evaluation

Regression

1. Scoring ($R^2, MSE, MAE, MAPE$)
2. Scatter Plot between Predicted & Actual Values

Code

Regression

- Scoring for **training set**

```
print('r2_score =\t\t\t', r2_score(y_train, y_pred_train))
print('mean_squared_error =\t\t', mean_squared_error(y_train, y_pred_train))
print('mean_absolute_error =\t\t', mean_absolute_error(y_train, y_pred_train))
print('mean_absolute_percentage_error =', mean_absolute_percentage_error(y_train, y_pred_train))
```

- Scoring for **test set**

```
print('r2_score =\t\t\t', r2_score(y_test, y_pred_test))
print('mean_squared_error =\t\t', mean_squared_error(y_test, y_pred_test))
print('mean_absolute_error =\t\t', mean_absolute_error(y_test, y_pred_test))
print('mean_absolute_percentage_error =', mean_absolute_percentage_error(y_test, y_pred_test))
```

Code

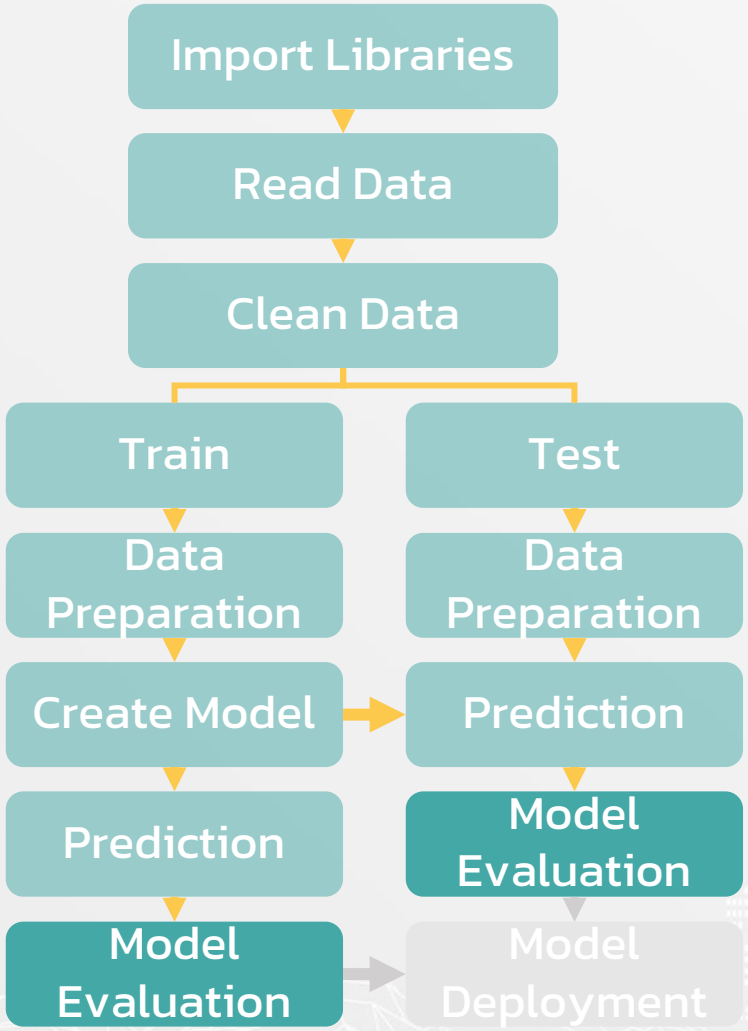
Regression

- Scatter Plot for **training set**

```
plt.scatter(y_pred_train, y_train)

plt.plot(y_pred_train, y_pred_train, color='red')

plt.title('Scatter Plot between Predicted & Actual Values')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

- Scatter Plot for **test set**

```
plt.scatter(y_pred_test, y_test)

plt.plot(y_pred_test, y_pred_test, color='red')

plt.title('Scatter Plot between Predicted & Actual Values')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```



Model Evaluation

Classification

1. Confusion Matrix
2. Scoring ($accuracy, precision, recall, F1$)

Code

Classification

- Confusion Matrix for **training set**

```
1 plot_confusion_matrix(clf, X_train_scaled, y_train)
```

- Confusion Matrix for **test set**

```
1 plot_confusion_matrix(clf, X_train_scaled, y_train)
```

Code

Classification

- Scoring for **training set**

```
1 report = classification_report(y_train, y_pred_train, output_dict=True)
```

```
1 print('accuracy =', report['accuracy'])
```

```
1 report['accept']
```

```
1 report['reject']
```

Code

Classification

- Scoring for **test set**

```
1 report = classification_report(y_test, y_pred_test, output_dict=True)
```

```
1 print('accuracy =', report['accuracy'])
```

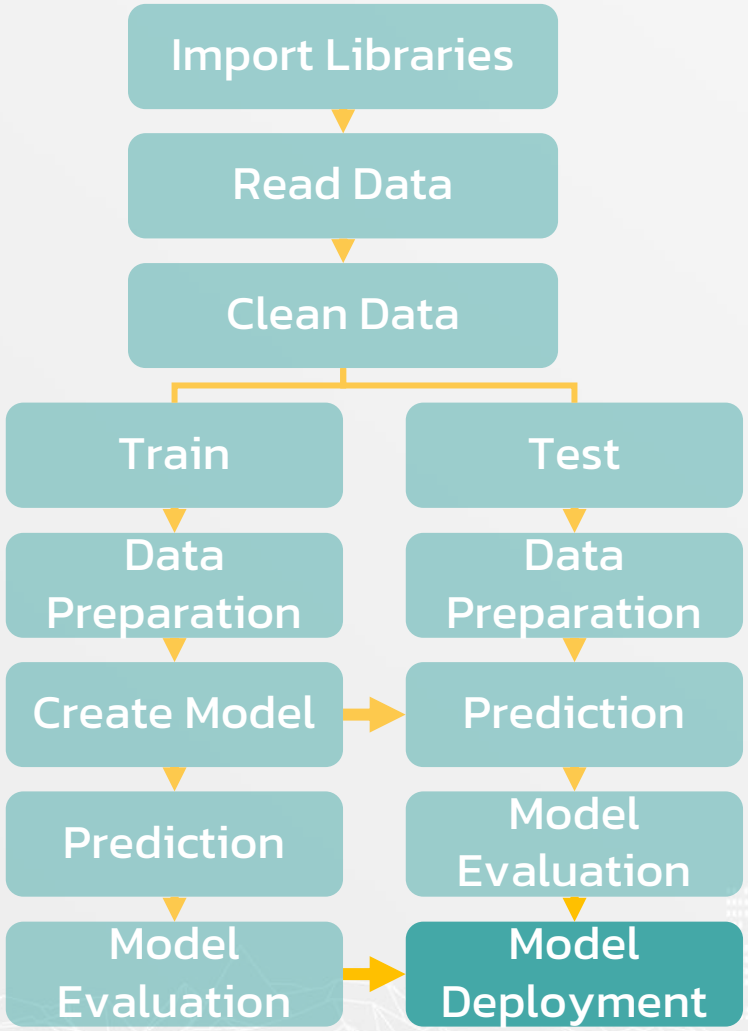
```
1 report['accept']
```

```
1 report['reject']
```

Model Deployment

Regression

Classification



Code

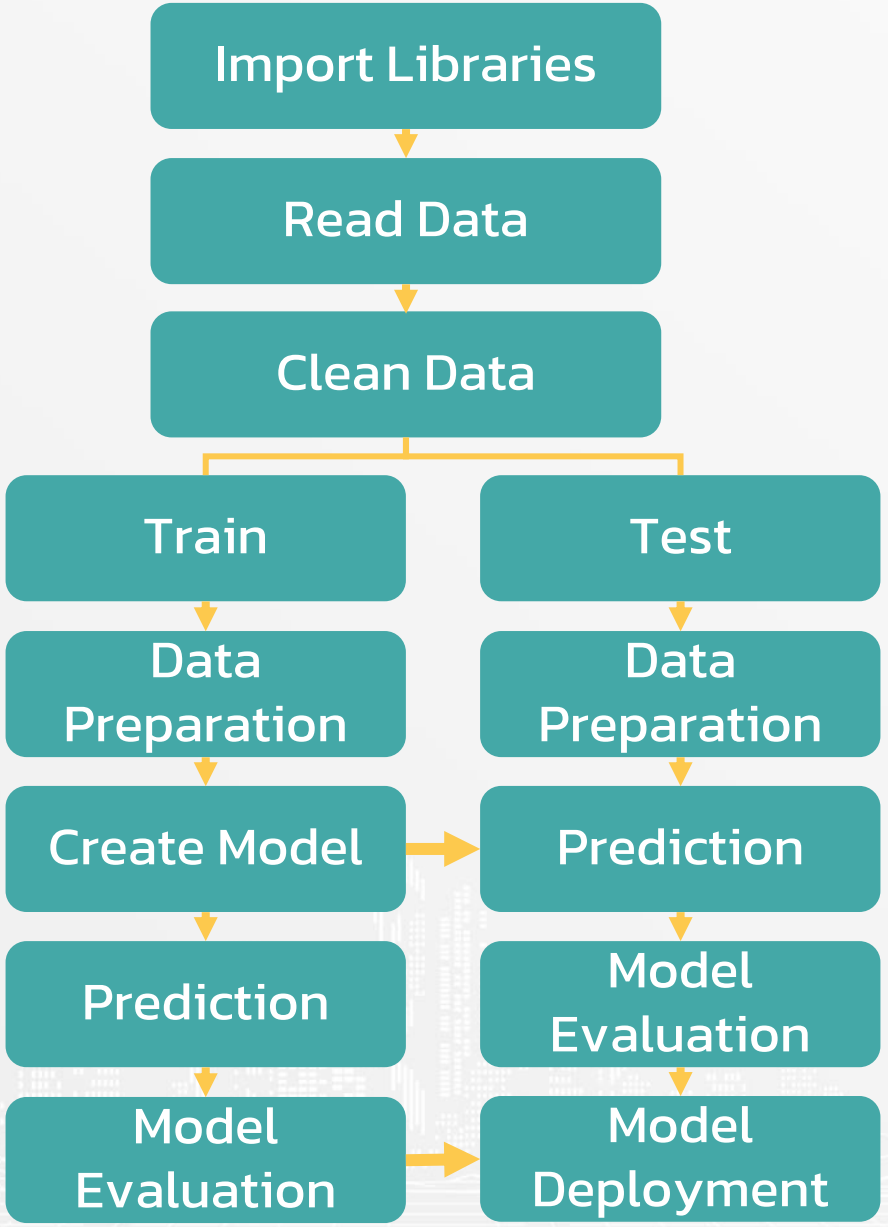
Regression

```
1  import pickle
2
3  pickle.dump((reg,
4               ordinal_encoder,
5               one_hot_encoder,
6               feature_name,
7               numerical_feature,
8               ordinal_feature,
9               nominal_feature),
10             open('salary_model.pickle', 'wb'))
```

Code

Classification

```python
import pickle

pickle.dump((clf,
             ordinal_encoder,
             one_hot_encoder,
             scaler,
             feature_name,
             numerical_feature,
             ordinal_feature,
             nominal_feature),
            open('job_acceptance_model.pickle', 'wb'))
```


Import Libraries
Read Data
Clean Data
Train
Test
Data Preparation
Data Preparation
Create Model
Prediction
Prediction
Model Evaluation
Model Evaluation
Model Deployment