DATA 301: Linear Regression review
Ordinary Least Squares (OLS)

Topics

Training Overview
OLS Linear Regression – outline
OLS Linear Regression - scikit-learn

Training a Model - Soon

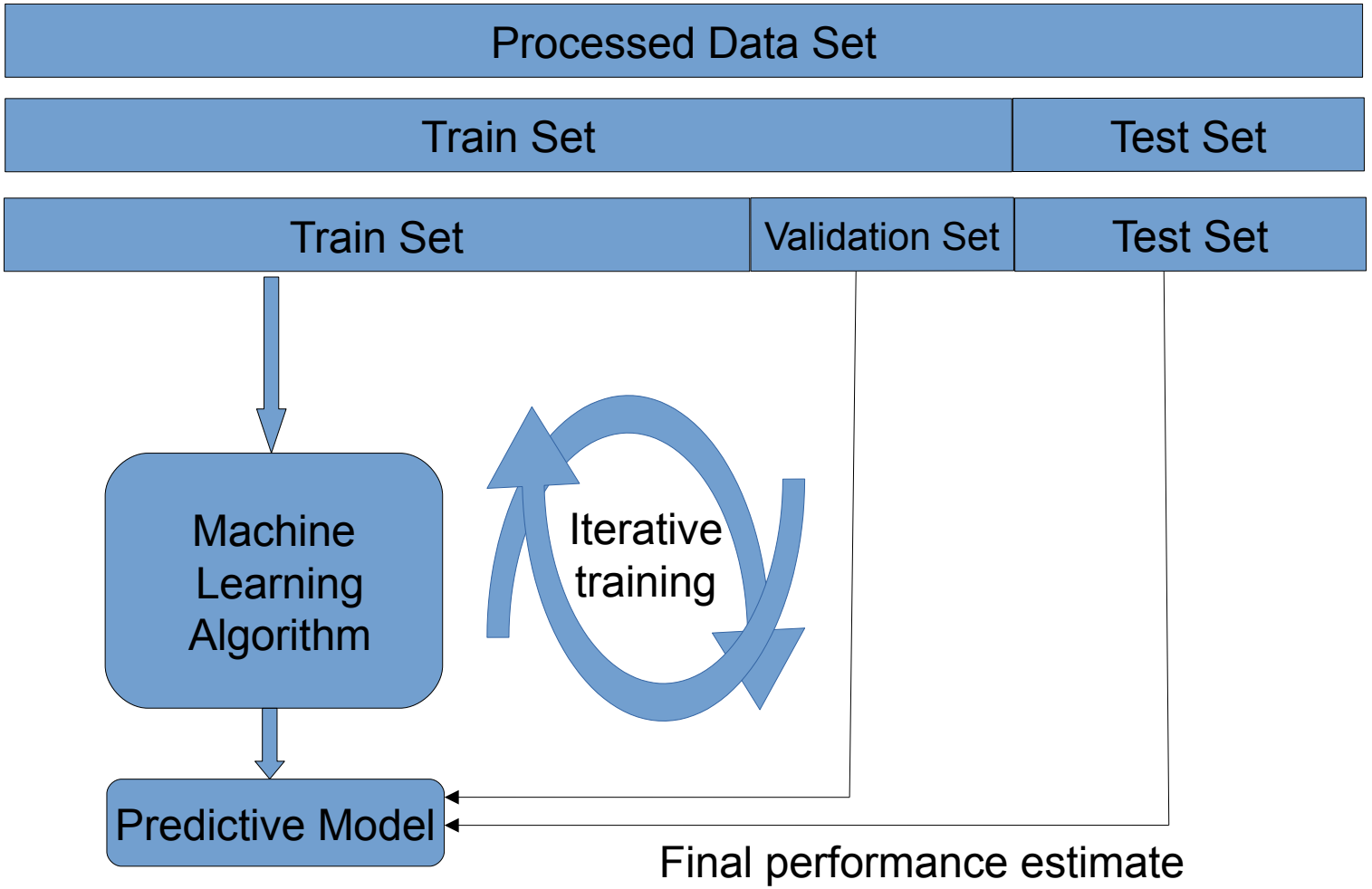


Figure from 'Python Machine Learning' by Sebastian Raschka

Training a Model - Soon

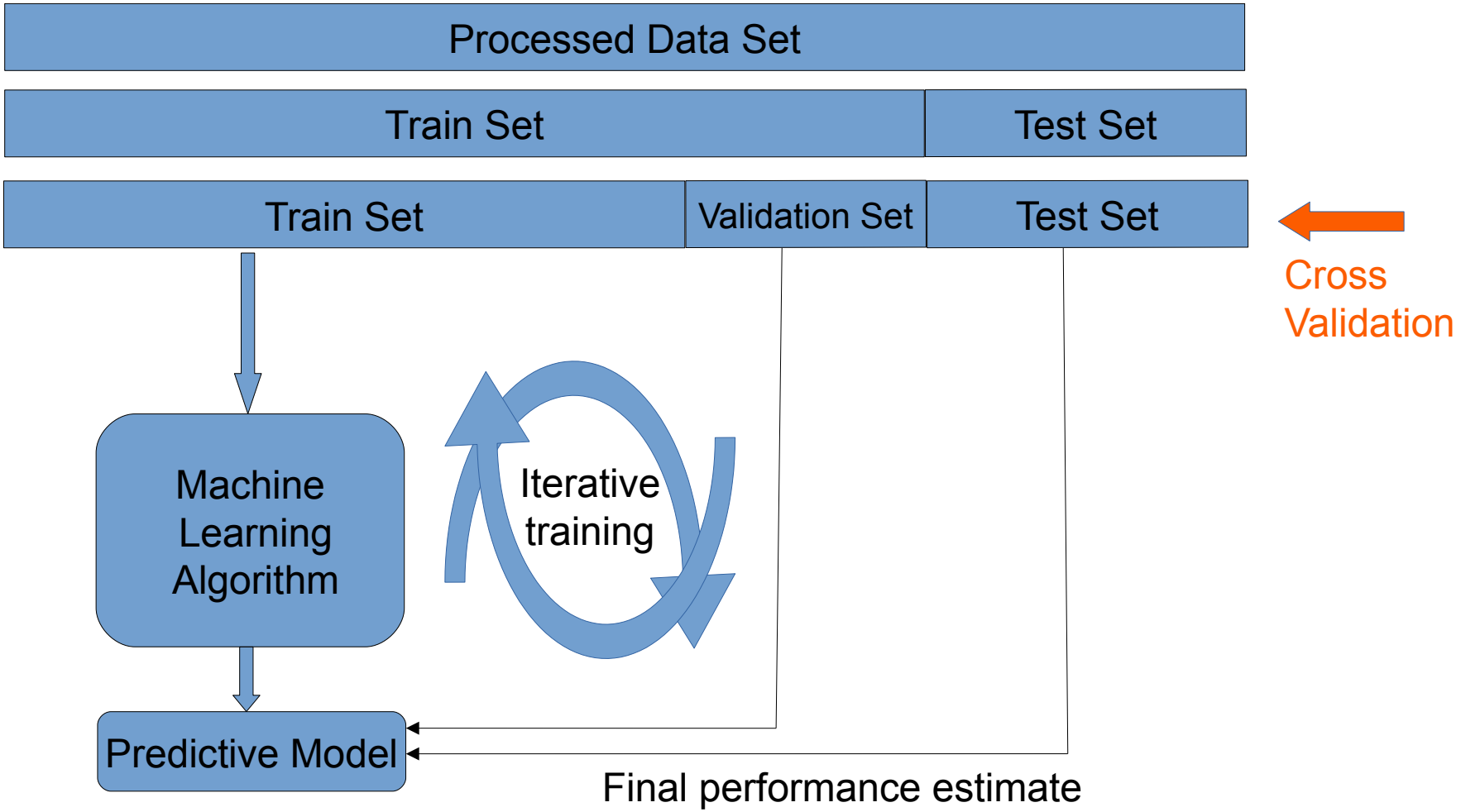


Figure from 'Python Machine Learning' by Sebastian Raschka

Training a Model - Today

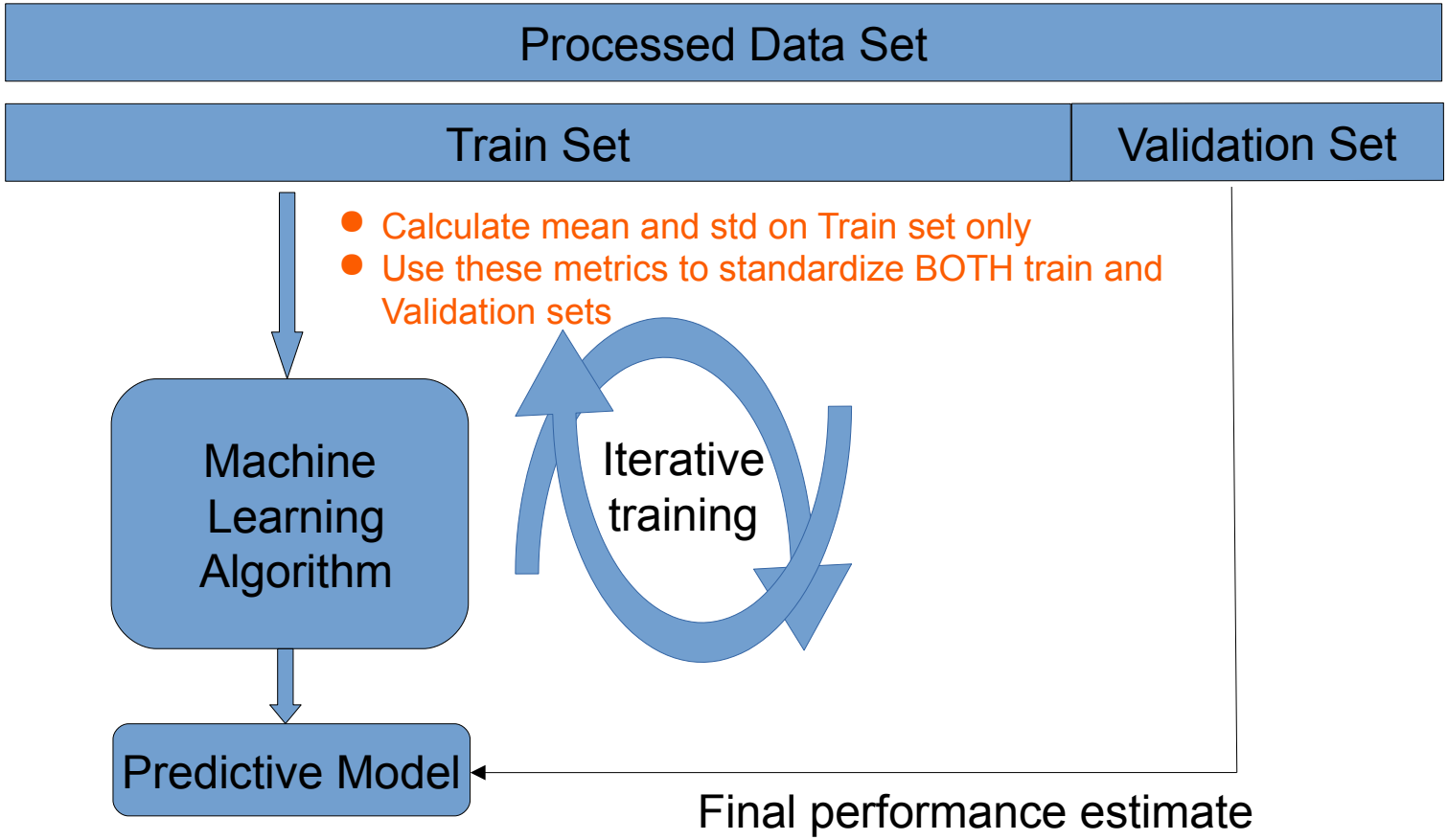
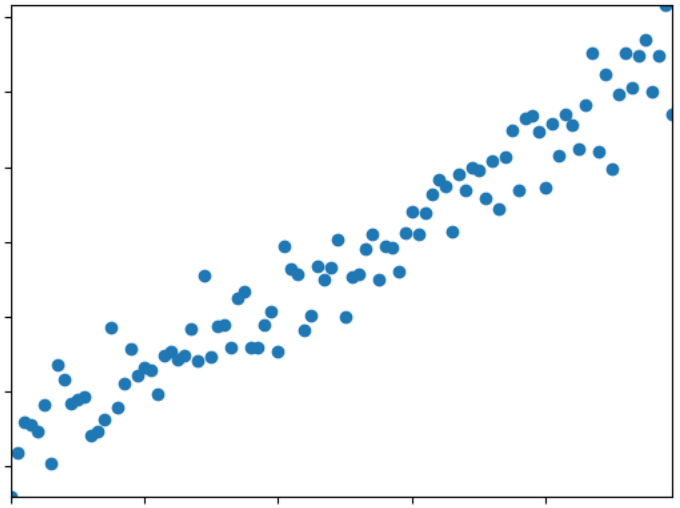


Figure from 'Python Machine Learning' by Sebastian Raschka

Training a Model

Ex. Linear regression iteratively estimates w terms in this equation

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$



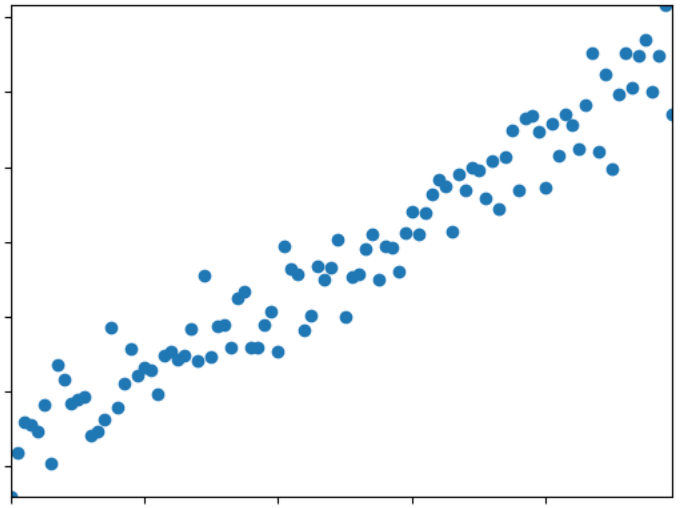
Training a Model

Ex. Linear regression iteratively estimates w terms in this equation

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

By reducing the error between actual and predicted values using this equation

$$\min_w ||Xw - y||_2^2$$



Training a Model

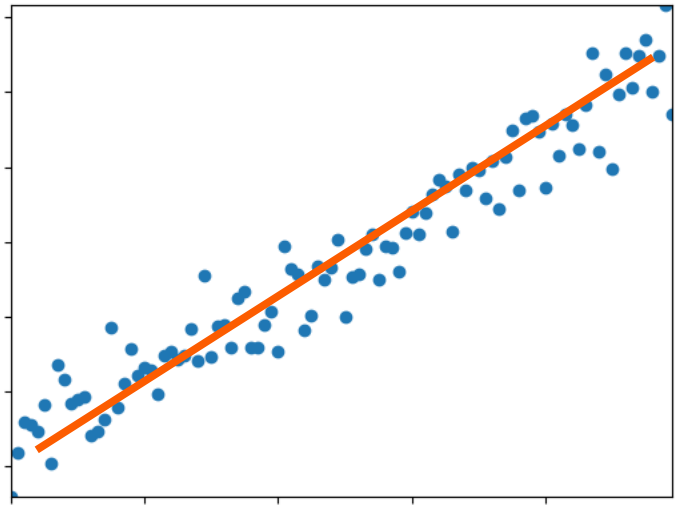
Ex. Linear regression iteratively estimates w terms in this equation

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

By reducing the error between actual and predicted values using this equation

$$\min_w ||Xw - y||_2^2$$

To generate a best fit line



Training a Model

```
#split into training and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

Split data

Training a Model

```
#split into training and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
#calculate mean and std on train set only!
scaler_Xtrain = preprocessing.StandardScaler().fit(X_train)
scaler_ytrain = preprocessing.StandardScaler().fit(y_train)
```

```
#then apply scaler params to train and test set (standardize)
X_train=scaler_Xtrain.transform(X_train)
y_train=scaler_ytrain.transform(y_train)
X_test=scaler_Xtrain.transform(X_test)
y_test=scaler_ytrain.transform(y_test)
```

Standardize data

- Calculate std and mean on train set only
- Use these metrics to standardize train and test sets

Training a Model

```
#split into training and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
#calculate mean and std on train set only!
scaler_Xtrain = preprocessing.StandardScaler().fit(X_train)
scaler_ytrain = preprocessing.StandardScaler().fit(y_train)
```

```
#then apply scaler params to train and test set (standardize)
X_train=scaler_Xtrain.transform(X_train)
y_train=scaler_ytrain.transform(y_train)
X_test=scaler_Xtrain.transform(X_test)
y_test=scaler_ytrain.transform(y_test)
```

```
from sklearn import linear_model

#create model
reg = linear_model.LinearRegression()

#fit model to data
reg.fit(X=X_train, y=y_train);
```

Create a model and fit it to train data, this is where linear regression parameters are calculated

Training a Model

```
#split into training and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
#calculate mean and std on train set only!
scaler_Xtrain = preprocessing.StandardScaler().fit(X_train)
scaler_ytrain = preprocessing.StandardScaler().fit(y_train)
```

```
#then apply scaler params to train and test set (standardize)
X_train=scaler_Xtrain.transform(X_train)
y_train=scaler_ytrain.transform(y_train)
X_test=scaler_Xtrain.transform(X_test)
y_test=scaler_ytrain.transform(y_test)
```

```
from sklearn import linear_model

#create model
reg = linear_model.LinearRegression()

#fit model to data
reg.fit(X=X_train, y=y_train);
```

```
#predict on new data
y_pred = reg.predict(X_test)
```

Use trained model to predict on unseen data
Linear regression parameters are fixed when predicting

Training a Model

```
#split into training and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
#calculate mean and std on train set only!
scaler_Xtrain = preprocessing.StandardScaler().fit(X_train)
scaler_ytrain = preprocessing.StandardScaler().fit(y_train)
```

```
#then apply scaler params to train and test set (standardize)
X_train=scaler_Xtrain.transform(X_train)
y_train=scaler_ytrain.transform(y_train)
X_test=scaler_Xtrain.transform(X_test)
y_test=scaler_ytrain.transform(y_test)
```

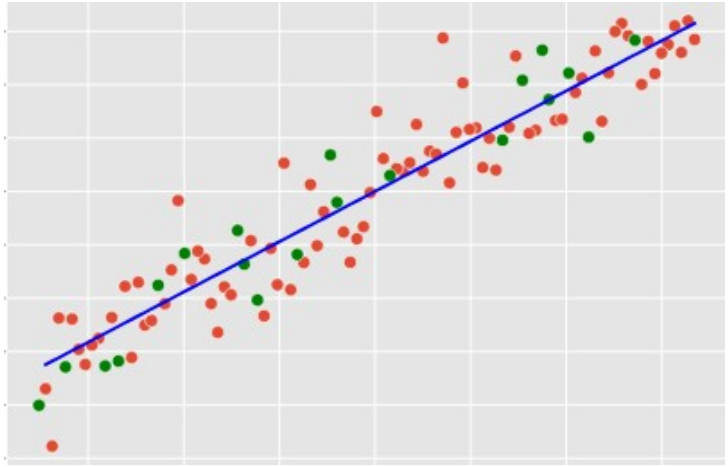
```
from sklearn import linear_model

#create model
reg = linear_model.LinearRegression()

#fit model to data
reg.fit(X=X_train, y=y_train);
```

```
#predict on new data
y_pred = reg.predict(X_test)
```

```
#plot points and linear regression line
ax=sns.scatterplot(x=X_train.squeeze(), y=y_train)
ax=sns.scatterplot(x=X_test.squeeze(), y=y_test, color='green')
ax=sns.lineplot(x=X_train.squeeze(), y=reg.predict(X_train),color='blue' )
```



Graphs;
train points
Regression line
Test points

Training a Model

```
#split into training and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
#calculate mean and std on train set only!
scaler_Xtrain = preprocessing.StandardScaler().fit(X_train)
scaler_ytrain = preprocessing.StandardScaler().fit(y_train)
```

```
#then apply scaler params to train and test set (standardize)
X_train=scaler_Xtrain.transform(X_train)
y_train=scaler_ytrain.transform(y_train)
X_test=scaler_Xtrain.transform(X_test)
y_test=scaler_ytrain.transform(y_test)
```

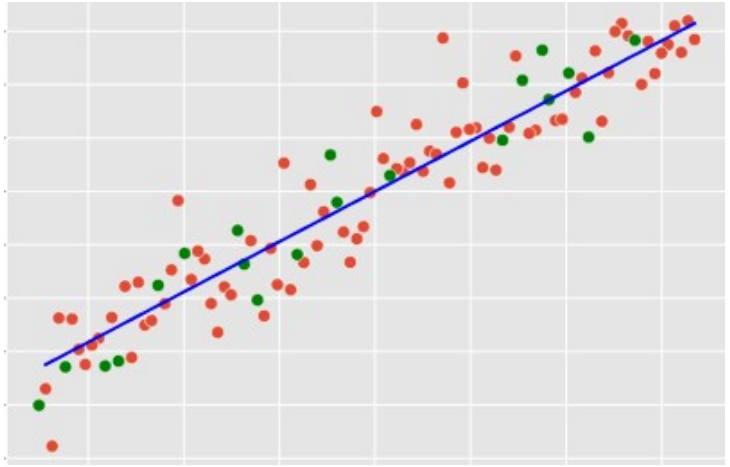
```
from sklearn import linear_model

#create model
reg = linear_model.LinearRegression()

#fit model to data
reg.fit(X=X_train, y=y_train);
```

```
#predict on new data
y_pred = reg.predict(X_test)
```

```
#plot points and linear regression line
ax=sns.scatterplot(x=X_train.squeeze(), y=y_train)
ax=sns.scatterplot(x=X_test.squeeze(), y=y_test, color='green')
ax=sns.lineplot(x=X_train.squeeze(), y=reg.predict(X_train),color='blue' )
```



```
from sklearn.metrics import mean_squared_error

#mean squared error?
print("Mean squared error for test: %.2f" % mean_squared_error(y_test, y_pred))
print("Mean squared error for train: %.2f" % mean_squared_error(y_train, reg.predict(X_train)))
```

```
Mean squared error for test: 0.09
Mean squared error for train: 0.11
```

And finally the errors for train and test sets

Summary

Standardize on train set only
Scikitlearn StandardScaler usage
Scikitlearn LinearRegression