- How do I use the pandas library to read data into python?
- How do I use the seaborn library to visualize data?
- What is linear regression, and how does it work?
- How do I train and interpret a linear regression model in scikit-learn?
- What are some evaluation metrices for regression problems?
- How do I choose which features to include in my model?

Types of supervised learning

- Classification: Predict a categorical response
- Regression: Predict a continuous response

```
import pandas as pd
data=pd.read_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv')
data.head()
```

#to set first col as index:

```
data=pd.read_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv',index_col=0)
```

```
data.tail()
data.shape
```

#Primary object types:

- DataFrame: rows and columns
- Series: a single column

What are the features?

- TV: advertising dollars spent on TV for a single product in a given market
- Radio: advertising dollars spent over radio
- Newspaper: Advertising dollars spent on newspaper

Respons:-

- Sales: Sales of a single product in a given market

What else:-

- Because the response variable is continuous, this is regression problem
- There are 200 observations and each observation is a single market

```
import seaborn as sns

%matplotlib inline

sns.pairplot(data,x_vars=['TV','radio','newspaper'],y_vars='sales’,size=7,aspect0.7,kind=’reg’)
```

Linear Regression is quite a deep topic but here we will discuss it in brief before starting with scikit learn.

Regression:- Type of supervised learning problem in which response is continuous

Linear Regression:- Machine learning model that can be used for regression problem

It runs quickly

No tuning required

Highly interpretable

Form of linear regression:-

$Y = a0 + a1x1 + a2x2 + \ldots\ldots + anxn$

Y:- Response

a0:- intercept

a1:-coefficient of x1(first feature)

an:-coefficient of x2(nth feature)

In this case:-

Y = a0 + a1 * TV + a2 * Radio + a3 * Newspaper

The “a” values are called the model coefficients. These values are “learned” during the model fitting step using the “least squares” criterion. Then, the fitted model can be used to make predictions.

Preparing X(feature matrix) and y(response) using pandas

- Scikit-learn expects X and y to be numpy array
- However, panda is built on top of numpy
- Thus X can be a pandas Dataframe and y can be a pandas Series

```
feature_cols=['TV','radio','newspaper']
X=data[feature_cols]
X=data[['TV','radio','newspaper']]
```

```
X.head()
```

```
type(X)
```

```
X.shape
```

```
y=data['sales']
y=data.sales
y.head()
```

Splitting X and Y into training and testing set

```
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1)
```

```
X_train.shape
y_train.shape
X_test.shape
y_test.shape
```

```
from sklearn.linear_model import LinearRegression
linreg=LinearRegression()
linreg.fit(X_train,y_train)
```

```
linreg.intercept_
```

```
linreg.coef_
```

y=2.88+0.0466*TV+0.179*Radio+0.00345*Newspaper

Make Predictions on testing sets

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

Evaluation Metrices:

1. Mean Absolute Error(MAE)

true=[100,50,30,20]

pred=[90+50+50+30]

$$\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$$

#calculate MAE:-

Manually:- (10+0+20+10)/4

Using Scikit learn:- from sklearn import metrics

print metrics.mean_absolute_error(true,pred)

2. Mean Squared Error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Manually:- (10**2+0**2+20**2+10**2)/4

Using scikit learn:-
metrices.mean_squared_error(true,pred)

3. Root Mean Squared Error (RMSE)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Manually:- import numpy as np

Print np.sqrt((10**2+0**2+20**2+10**2)/4)

Using scikit learn:- print np.sqrt(metrics.mean_squared_error(true,pred))

MAE:- Easiest to understand

MSE:- more popular as it punishes larger error

RMSE:- even more popular than MSE, as RMSE is interpretable in the “y” unit

Computing RMSE for our Sales Prediction:-

```
import numpy as np
from sklearn import metrics
np.sqrt(metrics.mean_squared_error(y_test,y_pred))
```

Feature Selection

#Does newspaper improve the quality of prediction

```
feature_cols=['TV','radio']
```

```
X=data[feature_cols]
y=data.sales
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1)
linreg.fit(X_train,y_train)
y_pred=linreg.predict(X_test)
np.sqrt(metrics.mean_squared_error(y_test,y_pred))
```