

LECTURE 2: SIMPLE LINEAR REGRESSION

Ehsan Aryafar

earyafar@pdx.edu

<http://web.cecs.pdx.edu/~aryafare/ML.html>

Recall: ML categories

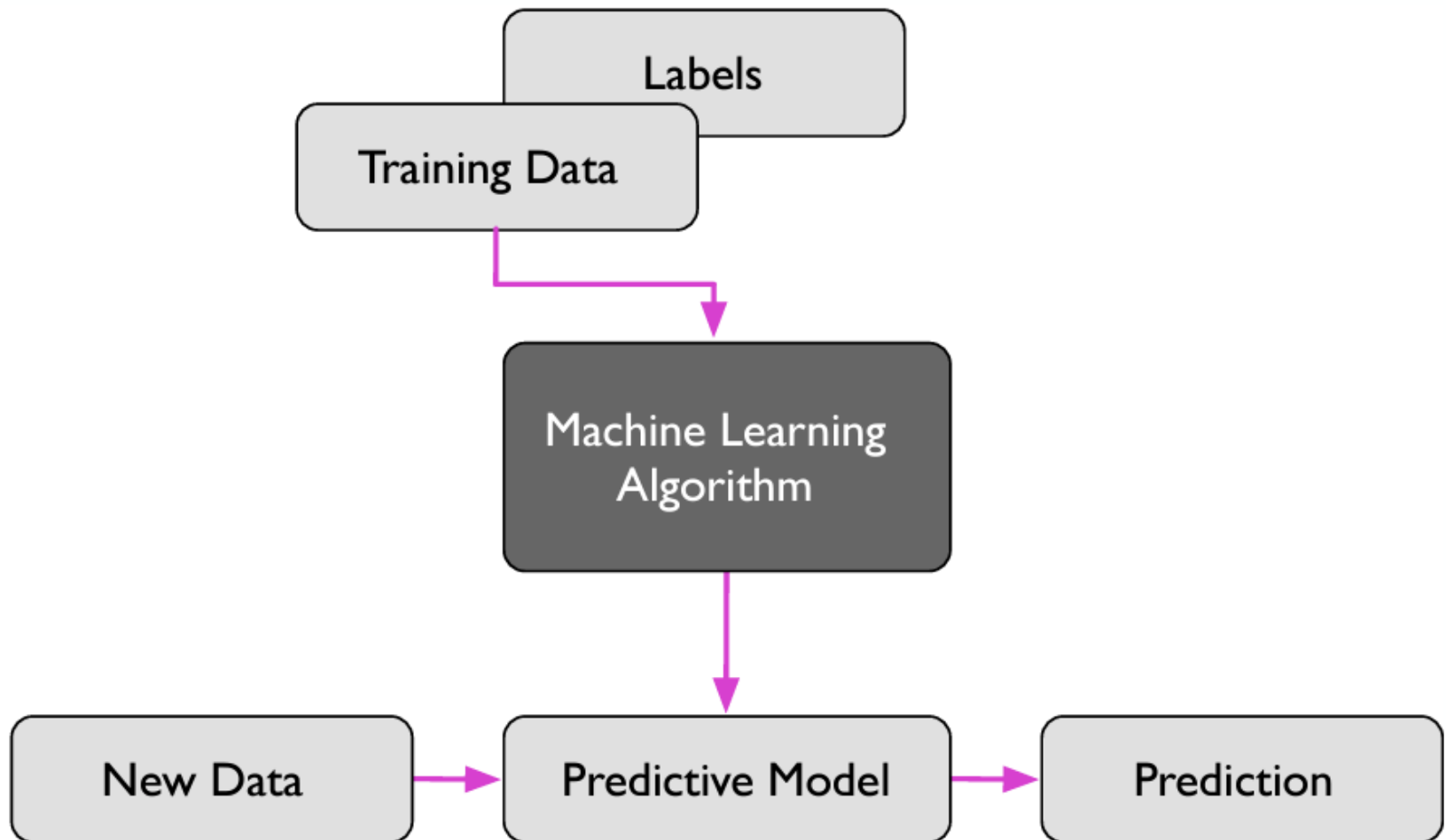
Supervised Learning

- Labeled data
- Direct feedback
- Predict outcome/future

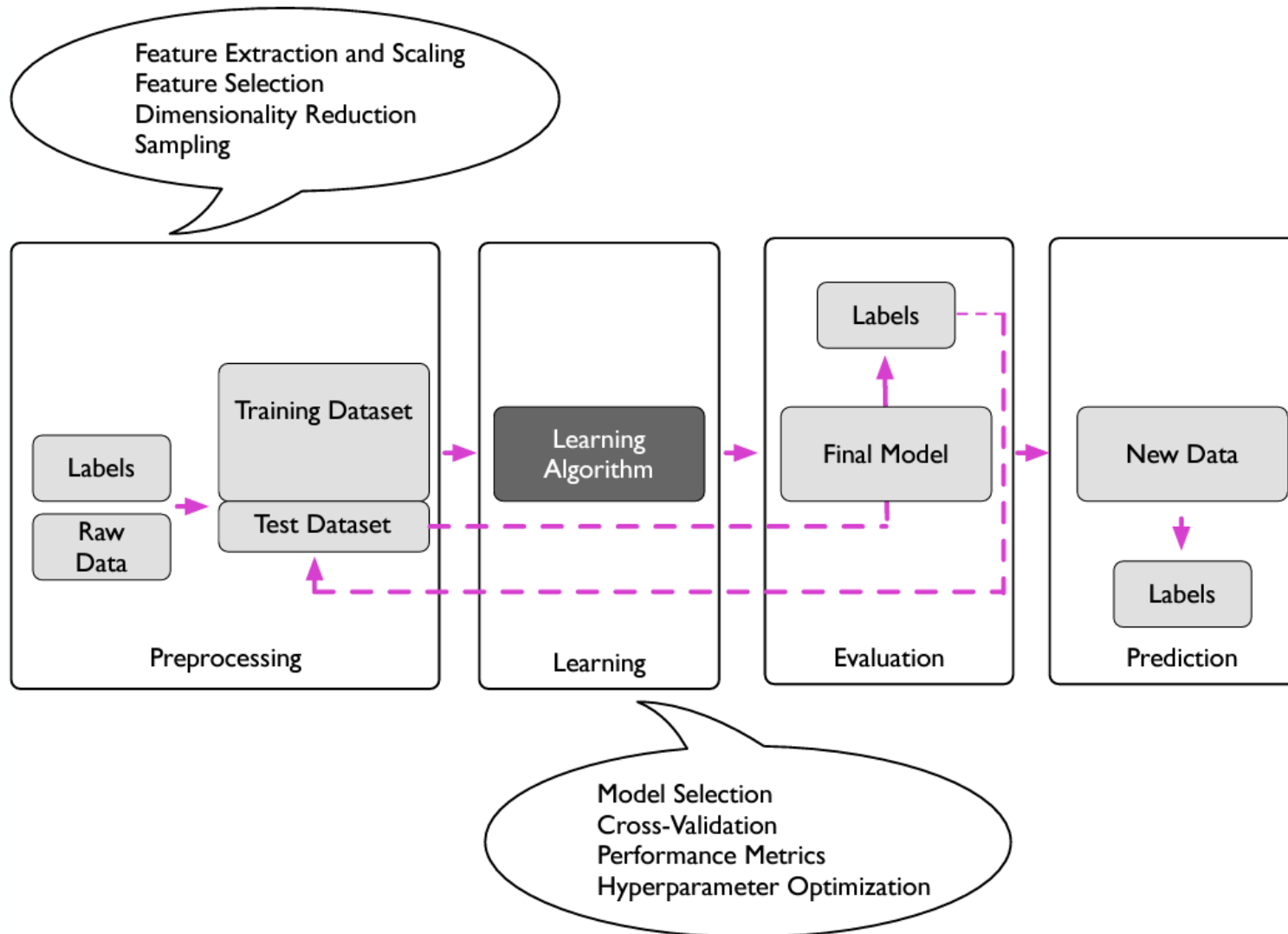
Unsupervised Learning

- No labels/targets
- No feedback
- Find hidden structure in data

Recall: Supervised Learning



Recall: Supervised Learning



Learning Objectives

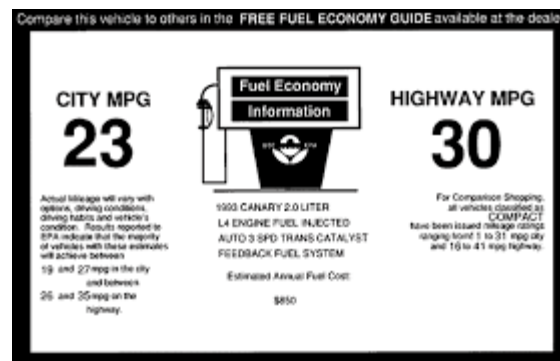
- How to load data from a text file
- How to visualize data via a scatter plot
- Describe a linear model for data
 - Identify the target variable and predictor
- Compute optimal parameters for the model using the regression formula
- Fit parameters for related models by minimizing the residual sum of squares
- Compute the R^2 measure of fit
- Visually determine goodness of fit and identify different causes for poor fit

Outline

- Motivating Example: Predicting the mpg of a car
- Linear Model
- Least Squares Fit Problem
- Sample Mean and Variance
- LS Fit Solution
- Assessing Goodness of Fit

Example: What Determines mpg in a Car?

- What engine characteristics determine fuel efficiency?
- Why would a data scientist be hired to answer this question?
- Not to help purchasing a specific car.
 - The mpg for a currently available car is already known.
 - If the car company isn't lying?
- To guide building new cars.
 - Understand what is reasonably achievable before full design



Run Code in Jupyter Notebook





Simple Linear Regression for Automobile mpg Data

In this demo, you will see how to:

- Load data from a text file using the `pandas` package
- Create a scatter plot of data
- Handle missing data
- Fit a simple linear model
- Plot the linear fit with the test data
- Use a nonlinear transformation for an improved fit (advanced concept)

Getting Data

- Data from UCI dataset library:
<https://archive.ics.uci.edu/ml/datasets.php>

 <u>Auto MPG</u>	Multivariate	Regression	Categorical, Real	398	8	1993
 <u>Automobile</u>	Multivariate	Regression	Categorical, Integer, Real	205	26	1987
 <u>Badges</u>	Univariate, Text	Classification		294	1	1994
 <u>Balance Scale</u>	Multivariate	Classification	Categorical	625	4	1994

Python Packages

- Python has many powerful packages or libraries

```
import pandas as pd
import numpy as np
```

- This demo uses three key packages

- Pandas:

- Used for reading and writing data files
- Loads data into dataframes

```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

- Numpy

- Numerical operations including linear algebra
- Data is stored in ndarray structure
- We convert from dataframes to ndarray

- Matplotlib:

- MATLAB-like plotting and visualization

Loading the Data in Jupyter Notebook

Try 1: The Wrong Way!

- Python pandas library
 - Read_csv command; Read URL or file location.

This creates a pandas *dataframe*. We can see the first six lines of the dataframe with `head` command:

```
[44]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data')
```

There were three errors:

- All the data appeared in one column. That is, the columns were not "delimited" correctly
- The first line got mistook as a header
- The columns are missing their header names

```
[45]: df.head(6)
```

```
[45]: 18.0 8 307.0 130.0 3504. 12.0 70 1\t"chevrolet chevelle malibu"
```

0	15.0 8 350.0 165.0 3693. 11...
1	18.0 8 318.0 150.0 3436. 11...
2	16.0 8 304.0 150.0 3433. 12...
3	17.0 8 302.0 140.0 3449. 10...
4	15.0 8 429.0 198.0 4341. 10...
5	14.0 8 454.0 220.0 4354. 9...

- Problems
- Does not parse columns
 - All data in a single column
 - Read_csv assumes columns are delimited by commas
- Mistakes first line as header

Try 2: Fixing the Errors

- Fix the arguments in `read_csv`

▼ Try 2: Fixing the Errors in the loading



The problems above are common. Often it takes a few times to load the data correctly. That is why it is good to look at the first few elements of the dataframe before proceeding. After some googling you can find out that you need to specify some other options to the `read_csv` command. First, you need to supply the names of the columns. In this case, I have supplied them manually based on the description in the UCI website:

```
[46]: names = ['mpg', 'cylinders', 'displacement', 'horsepower',  
             'weight', 'acceleration', 'model year', 'origin', 'car name']  
  
[47]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/'+  
                      'auto-mpg/auto-mpg.data',  
                      header=None, delim_whitespace=True, names=names, na_values='?')
```

If you re-run `head` command now, you can see the loading was correct. You can see the column names, index, and values:

Try 2: Fixing the Errors

Try 2: Fixing the Errors in the loading



The problems above are common. Often it takes a few times to load the data correctly. That is why it is good to look at the first few elements of the dataframe before proceeding. After some googling you can find out that you need to specify some other options to the `read_csv` command. First, you need to supply the names of the columns. In this case, I have supplied them manually based on the description in the UCI website:

```
[46]: names = ['mpg', 'cylinders', 'displacement', 'horsepower',
              'weight', 'acceleration', 'model year', 'origin', 'car name']

[47]: df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/'+
                      'auto-mpg/auto-mpg.data',
                      header=None, delim_whitespace=True, names=names, na_values='?')
```

If you re-run `head` command now, you can see the loading was correct. You can see the column names, index, and values:

```
[49]: df.head(6)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino
5	15.0	8	429.0	198.0	4341.0	10.0	70	1	ford galaxie 500

A Dataframe Object

- A dataframe object is stored in a table and has three components

- `df.columns`: returns the names of the columns

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',  
      'acceleration', 'model year', 'origin', 'car name'],  
      dtype='object')
```

- `df.index`: returns the indices of the rows

```
df.index
```

```
RangeIndex(start=0, stop=398, step=1)
```

- `df.values` is a 2-D array with values of the attributes for each car

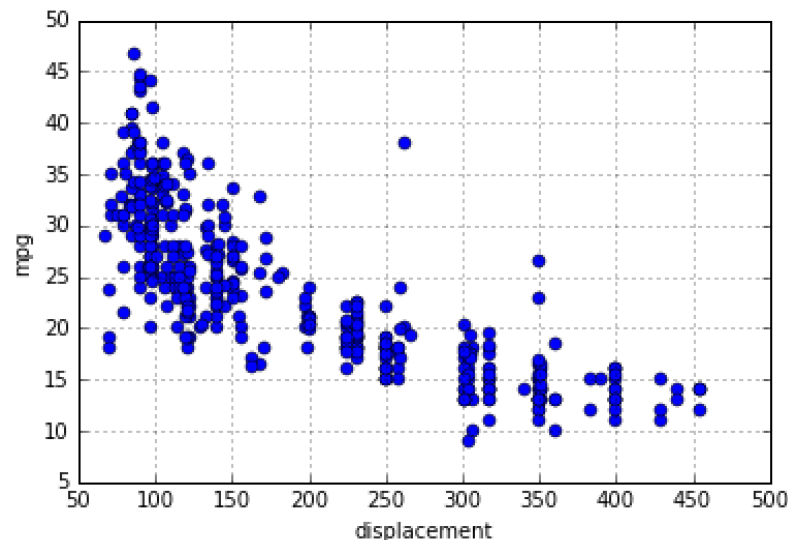
Visualizing the Data

```
In [150]: xstr = 'displacement'  
x = np.array(df[xstr])  
y = np.array(df['mpg'])
```

```
In [146]: import matplotlib  
import matplotlib.pyplot as plt  
%matplotlib inline
```

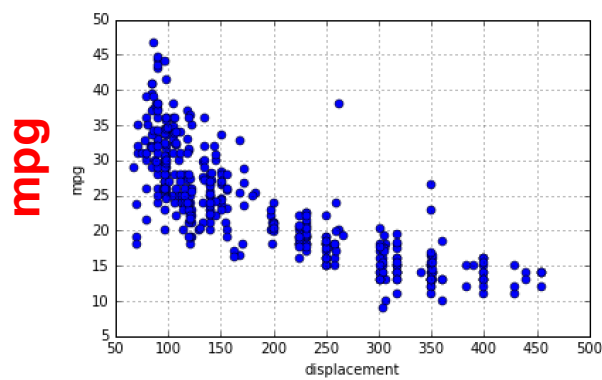
```
In [151]: plt.plot(x,y,'o')  
plt.xlabel(xstr)  
plt.ylabel('mpg')  
plt.grid(True)
```

- When possible, look at data before doing anything
- Python has MATLAB-like plotting
 - Matplotlib module
- First we convert dataframes to numpy arrays

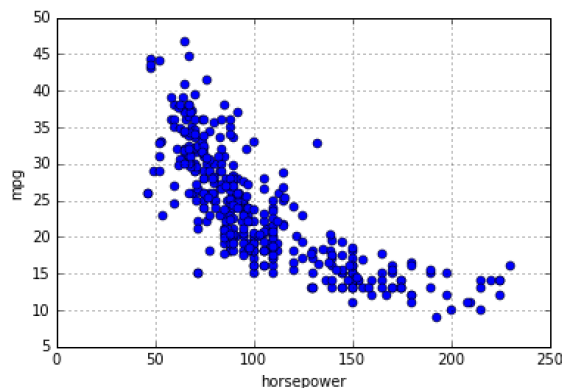


Exercise: Postulate a Model

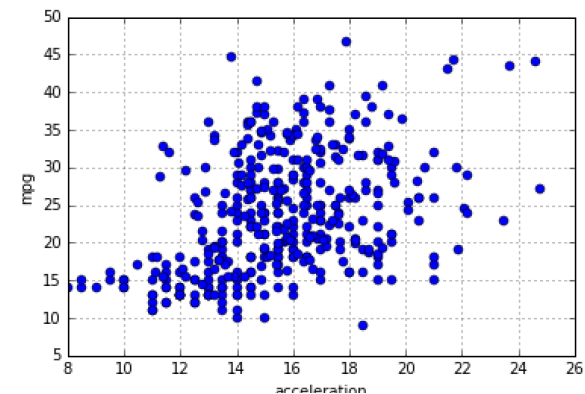
- Try to find a mathematical model to predict mpg from displacement, horsepower or acceleration
 - Make a reasonable / eyeball guess. No need for program now.
- What does your model predict when displacement = 200?
- Is the prediction reasonable? Can you improve your model?



Displacement



Horsepower



Acceleration

Outline

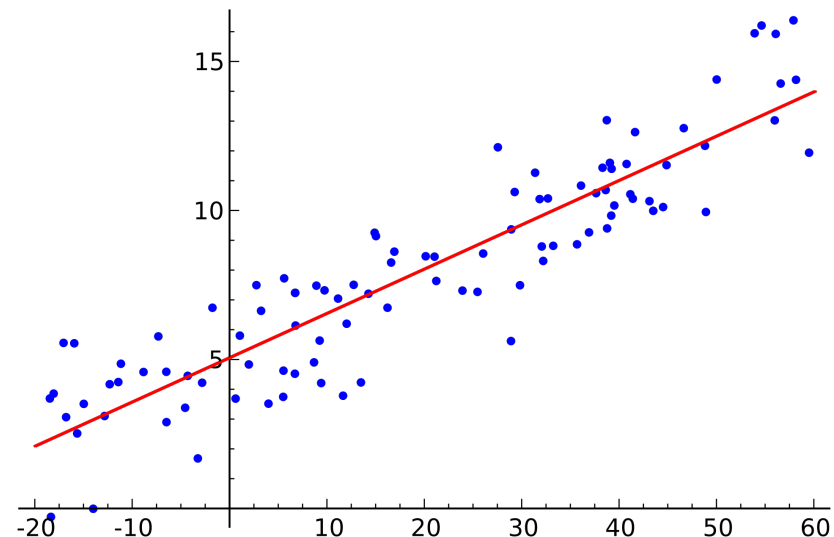
- Motivating Example: Predicting the mpg of a car
- Linear Model
- Least Squares Fit Problem
- Sample Mean and Variance
- LS Fit Solution
- Assessing Goodness of Fit

Linear Model Residual

- Knowing x does not exactly predict y
 - Variation in y due to factors other than x
- Add a **residual** term

$$y = \beta_0 + \beta_1 x + \epsilon$$

- **Residual = component the model does not explain**
 - Predicted value: $\hat{y}_i = \beta_1 x_i + \beta_0$
 - Residual: $\epsilon_i = y_i - \hat{y}_i$
 - *Sort of like residual error*
- Vertical deviation from the regression line
- β_0 is called **intercept** or **bias** and β_1 is called **slope** or **weight**



y_i is the target variable, which is the ground truth

\hat{y}_i is the outcome of prediction

Least Squares Model Fitting

- How do we select parameters $\beta = (\beta_0, \beta_1)$?
- Define $\hat{y}_i = \beta_1 x_i + \beta_0$
 - Predicted value on sample i for parameters $\beta = (\beta_0, \beta_1)$
- Define average residual sum of squares:

$$\text{RSS}(\beta_0, \beta_1) := \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- Note that \hat{y}_i is implicitly a function of $\beta = (\beta_0, \beta_1)$
 - Also called the sum of squared residuals (SSR) and sum of squared errors (SSE)
- Least squares solution: Find (β_0, β_1) to minimize RSS.
 - Geometrically, minimizes squared distances of samples to regression line

Finding Parameters via Optimization

A general ML recipe

General ML problem

- Find a model with parameters
- Get data
- Pick a loss function
 - Measures goodness of fit model to data
 - Function of the parameters
- Find parameters that minimizes loss

Simple linear regression

Linear model: $\hat{y} = \beta_0 + \beta_1 x$

Data: $(x_i, y_i), i = 1, 2, \dots, N$

Loss function:

$$RSS(\beta_0, \beta_1) := \sum (y_i - \beta_0 - \beta_1 x_i)^2$$

Select β_0, β_1 to minimize $RSS(\beta_0, \beta_1)$

Outline

- Motivating Example: Predicting the mpg of a car
- Linear Model
- Least Squares Fit Problem
- Sample Mean and Variance
- LS Fit Solution
- Assessing Goodness of Fit

Sample Mean and Standard Deviations

- Given data $(x_i, y_i), i = 1, \dots, N$

- Sample mean

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$$

- Sample variances

$$s_x^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2, \quad s_y^2 = \frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2$$

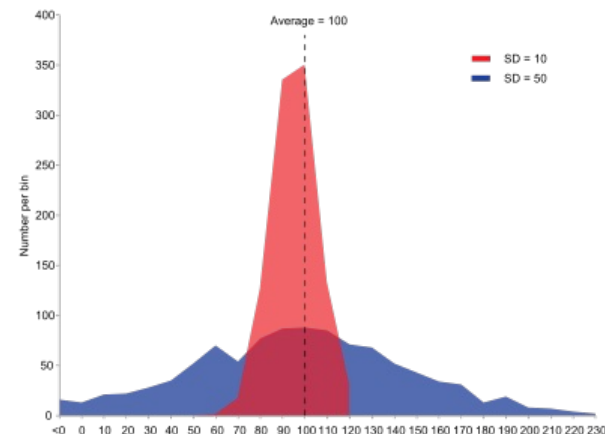
- For technical reasons, above formulae are called the biased variances.

- Sample standard deviation

- s_x, s_y (note this notation)**

- Square root of variances

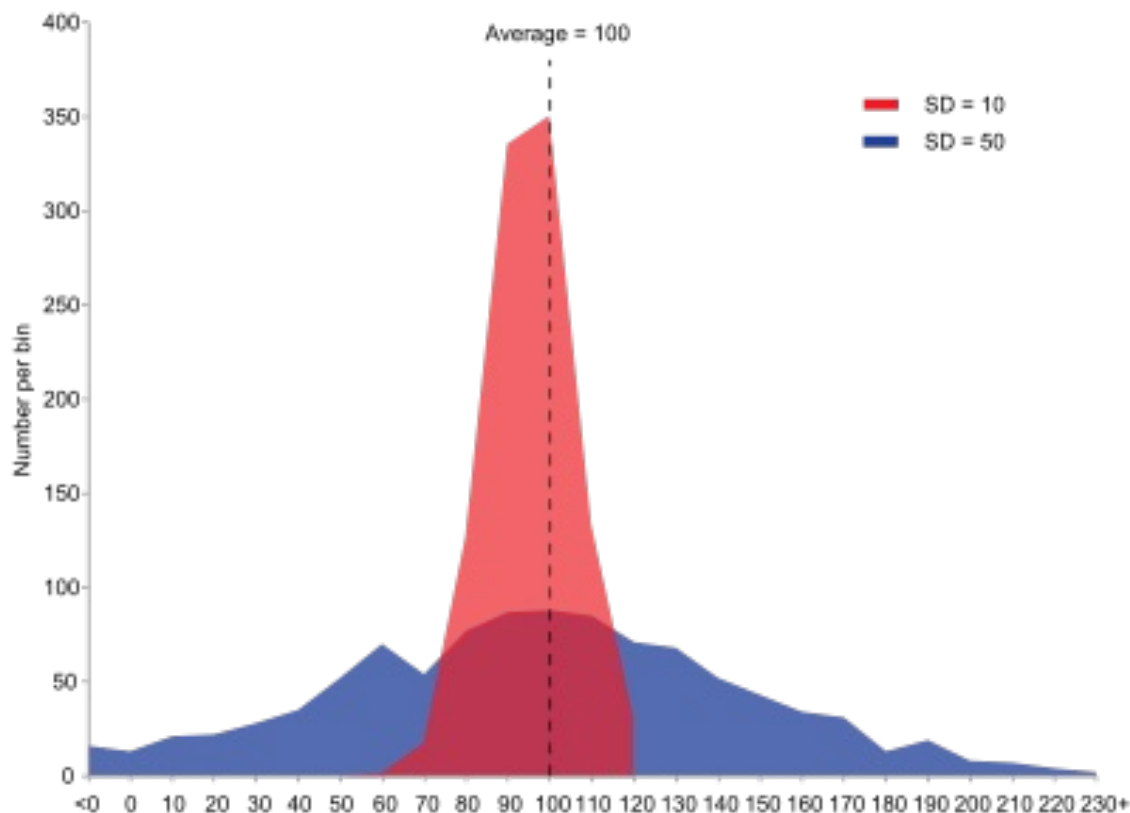
- Many times you drop sample from the term and just call it standard deviation



Visualizing standard deviation

https://en.wikipedia.org/wiki/Standard_deviation

Example Visualization of Standard Deviation



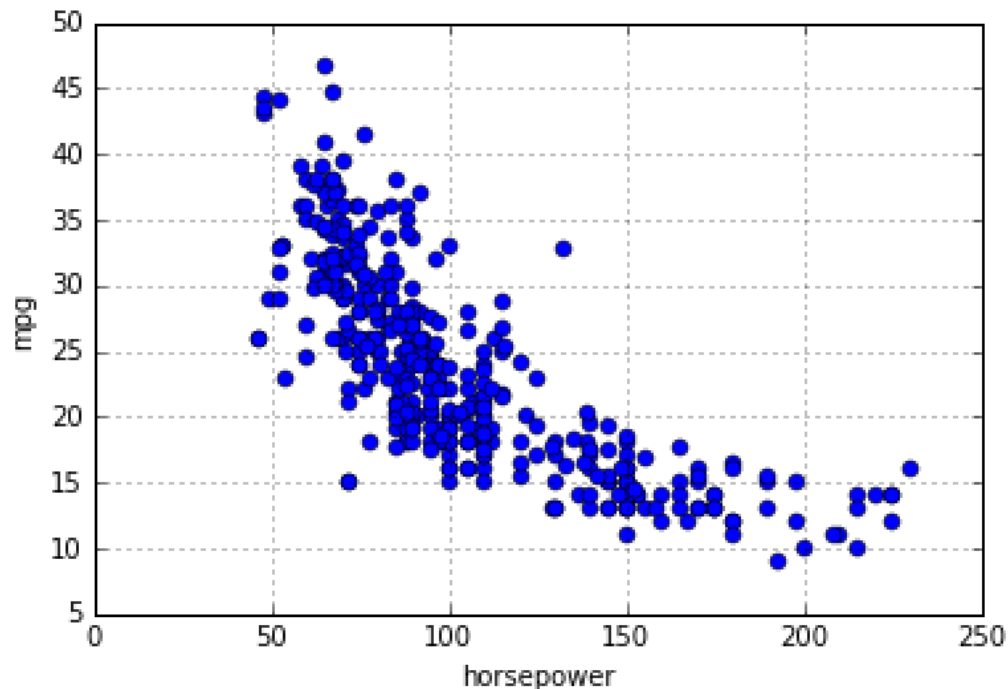
Visualizing standard deviation

https://en.wikipedia.org/wiki/Standard_deviation

Visualizing Mean and SD on Scatter Plot

Using the picture only (no calculators), estimate the following (roughly):

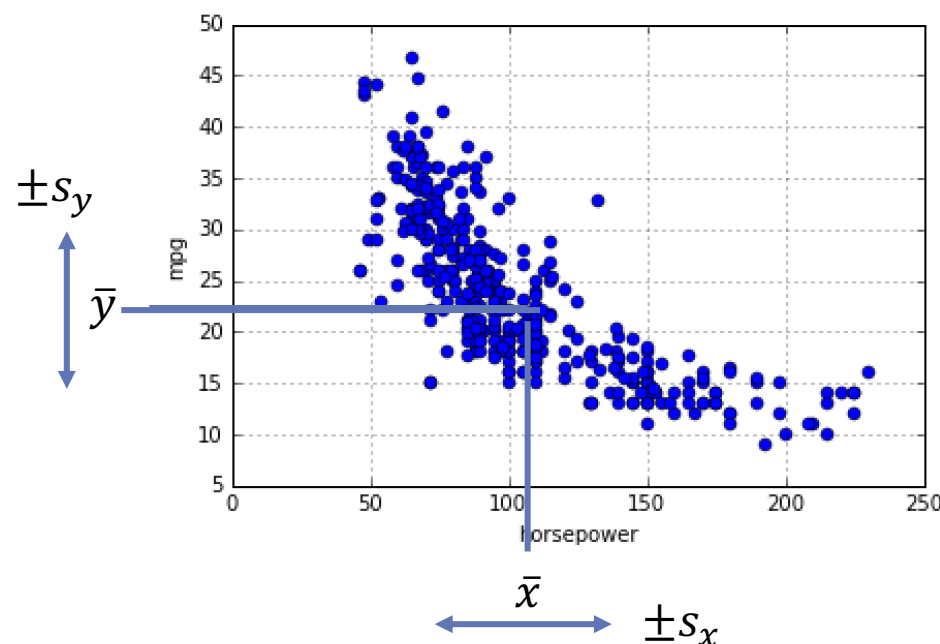
- The sample mean mpg and horsepower: \bar{x} , \bar{y}
- The sample std deviations: s_x , s_y



Visualizing Mean and SD on Scatter Plot

Approximate answer

- **Means:** \bar{x} and \bar{y}
 - Weighted center of the points in each axis
- **Standard deviations:** s_x and s_y
 - Represents “variation” in each axis from mean
 - With Gaussian distributions: 0.95% of points are 2 SDs from mean
 - What is a distribution (future lecture)

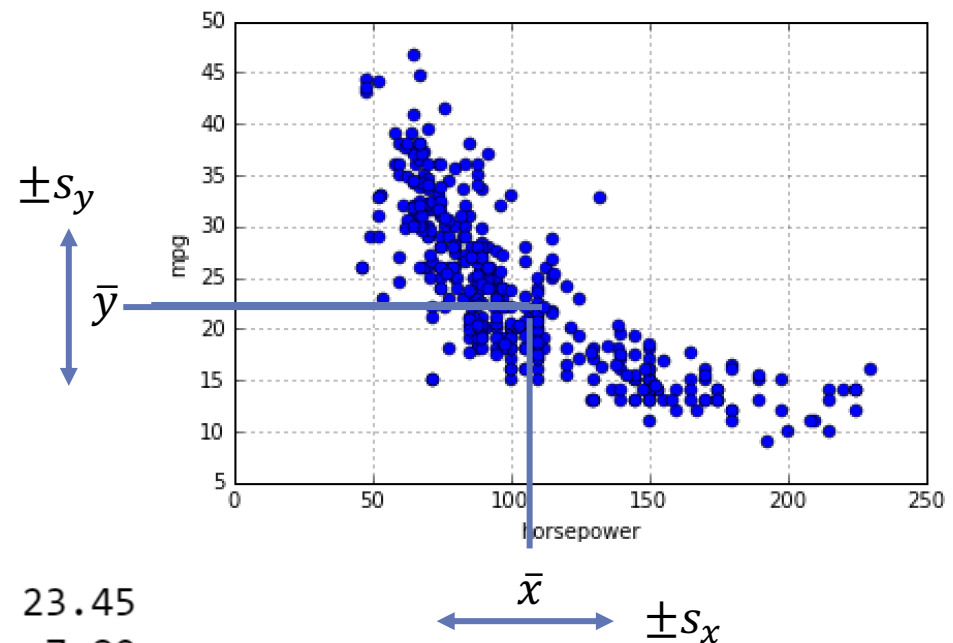


Computing Means and SD in Python

- Exact answer can be computed in python

```
xm = np.mean(x)
ym = np.mean(y)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
sxx = np.mean((x-xm)**2)
```

```
xbar      = 104.47,      ybar= 23.45
sqrt(sxx)= 38.44,  sqrt(syy)= 7.80
```



Sample Covariance

- Sample covariance:

$$s_{xy} = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

- Will interpret this momentarily
- Cauchy-Schwarz Law: $|s_{xy}| < s_x s_y$
- Sample correlation coefficient

$$r_{xy} = \frac{s_{xy}}{s_x s_y} \in [-1, 1]$$

Statistics

- Often need to compute averages of other functions of data
- **Definition:** The sample mean of a function $g(x, y)$ is:

$$\langle g(x_i, y_i) \rangle := \frac{1}{N} \sum_{i=1}^N g(x_i, y_i)$$

- Represents the average of $g(x, y)$ on the data
- Function $g(x, y)$ is called a **statistic**
- With this notation:
 - $\bar{x} = \langle x_i \rangle$, $\bar{y} = \langle y_i \rangle$
 - $s_{xx} = \langle (x_i - \bar{x})^2 \rangle$, $s_{yy} = \langle (y_i - \bar{y})^2 \rangle$

Alternate Equation for Variance

- **Alternate equations** for variance and sample co-variance:
 - Sample variances $s_{xx} = \langle x_i^2 \rangle - \langle x_i \rangle^2$, $s_{yy} = \langle y_i^2 \rangle - \langle y_i \rangle^2$
 - Sample co-variance $s_{xy} = \langle x_i y_i \rangle - \langle x_i \rangle \langle y_i \rangle$
- **Proof:**
 - $s_{xx} = \frac{1}{N} \sum (x_i - \bar{x})^2 = \frac{1}{N} \sum (x_i^2 - 2x_i\bar{x} + \bar{x}^2) = \langle x_i^2 \rangle - 2\bar{x}\langle x_i \rangle + \bar{x}^2$
 - Recall $\bar{x} = \langle x_i \rangle$
 - Therefore, $s_{xx} = \langle x_i^2 \rangle - \langle x_i \rangle^2$
 - Other relations $s_{yy} = \langle y_i^2 \rangle - \langle y_i \rangle^2$ and $s_{xy} = \langle x_i y_i \rangle - \langle x_i \rangle \langle y_i \rangle$ proved similarly

Notation

- This class will use the following notation
- We will try to be consistent
- Note: Other texts use different notations

Statistic	Notation	Formula	Python
Sample mean	\bar{x}	$\frac{1}{n} \sum_{i=1}^n x_i$	xm
Sample variance	$s_x^2 = s_{xx}$	$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$	sxx
Sample standard deviation	$s_x = \sqrt{s_{xx}}$	$s_x = \sqrt{s_{xx}}$	sx
Sample covariance	s_{xy}	$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$	sxy

```

xm = np.mean(x)
ym = np.mean(y)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
sxx = np.mean((x-xm)**2)

```

Outline

- Motivating Example: Predicting the mpg of a car
- Linear Model
- Least Squares Fit Problem
- Sample Mean and Variance
- LS Fit Solution
- Assessing Goodness of Fit

Minimizing RSS

- To minimize $RSS(\beta_0, \beta_1)$ take partial derivatives:

$$\frac{\partial RSS}{\partial \beta_0} = 0, \quad \frac{\partial RSS}{\partial \beta_1} = 0$$

- Taking derivatives we get two conditions (proof removed):

$$\sum_{i=1}^N \epsilon_i = 0, \quad \sum_{i=1}^N x_i \epsilon_i = 0 \quad \text{where } \epsilon_i = y_i - \beta_0 - \beta_1 x_i$$

- Regression equation:

- After some manipulation, solution to optimal slope and intercept:

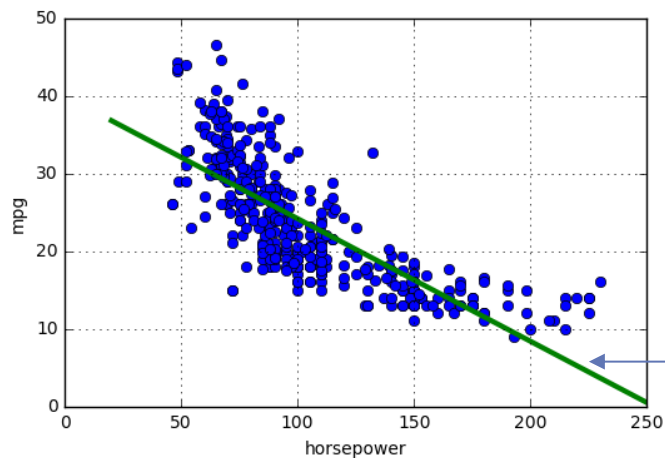
$$\beta_1 = \frac{s_{xy}}{s_x^2} = \frac{r_{xy} s_y}{s_x}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

Correlation coefficient between
x and y

Auto Example

- Python code



```
xm = np.mean(x)
ym = np.mean(y)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
sxx = np.mean((x-xm)**2)
beta1 = syx/sxx
beta0 = ym - beta1*xm
```

beta0= 39.94, beta1= -0.16

Regression line:

$$\text{mpg} = \beta_0 + \beta_1 \text{ horsepower}$$

Outline

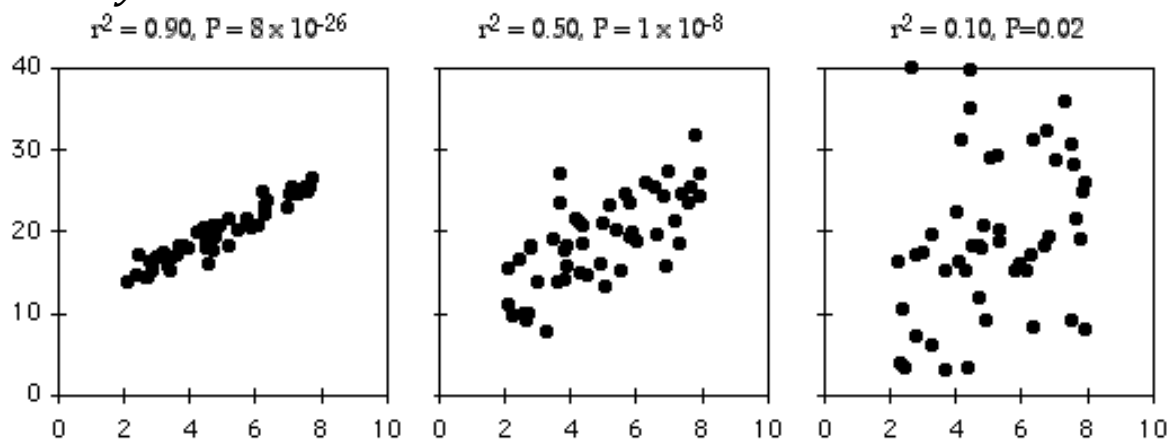
- Motivating Example: Predicting the mpg of a car
- Linear Model
- Least Squares Fit Problem
- Sample Mean and Variance
- LS Fit Solution
- Assessing Goodness of Fit

Minimum RSS

- Minimum RSS (Proof removed)

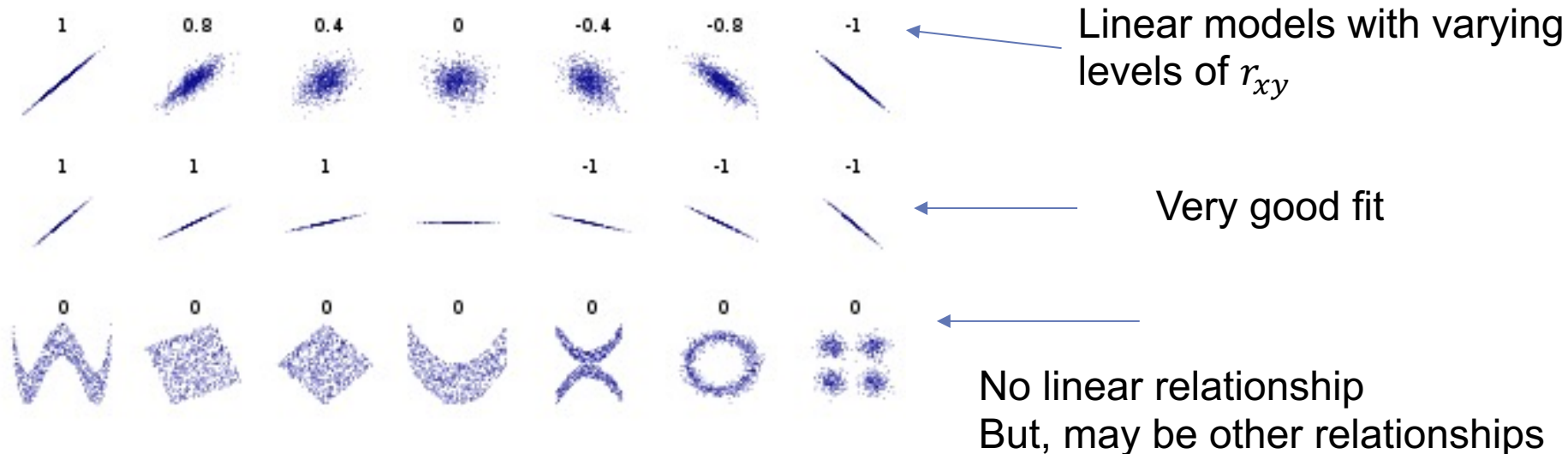
$$\min_{\beta_0, \beta_1} \text{RSS}(\beta_0, \beta_1) = N(1 - r_{xy}^2)s_y^2$$

- **Coefficient of Determination:** $R^2 = r_{xy}^2$
 - R^2 close to one makes RSS Close to 0
 - Explains portion of variance in y explained by x
 - s_y^2 =variance in target y
 - $(1 - R^2)s_y^2$ =residual sum of squares after accounting for x

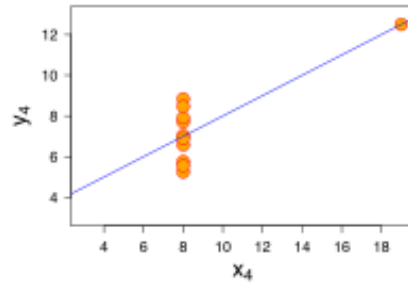
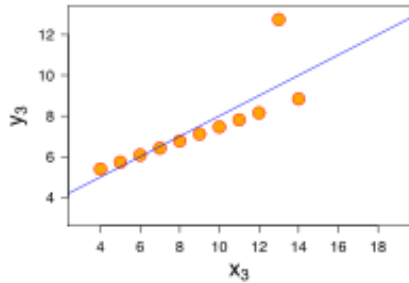
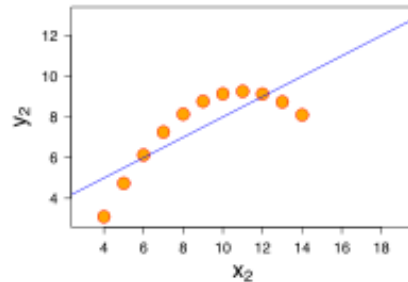
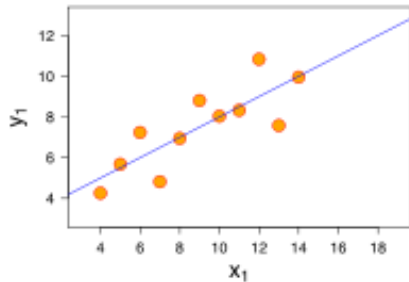


Visually seeing correlation

- $R^2 = r_{xy}^2 \approx 1$: Linear model is a very good fit
- $R^2 = r_{xy}^2 \approx 0$: Linear model is a poor fit.
- $\beta_1 = \frac{r_{xy}s_y}{s_x} \Rightarrow \text{Sign}(\beta_1) = \text{Sign}(r_{xy})$



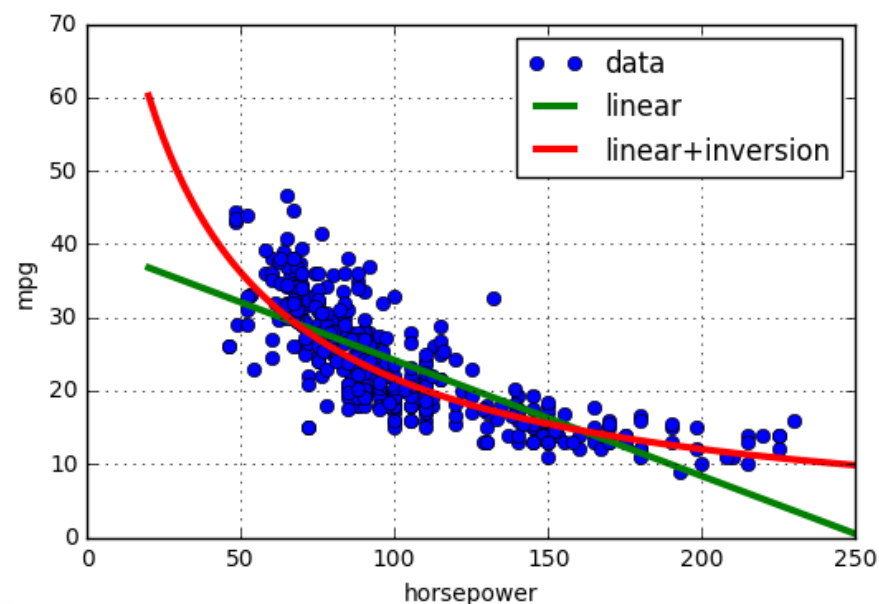
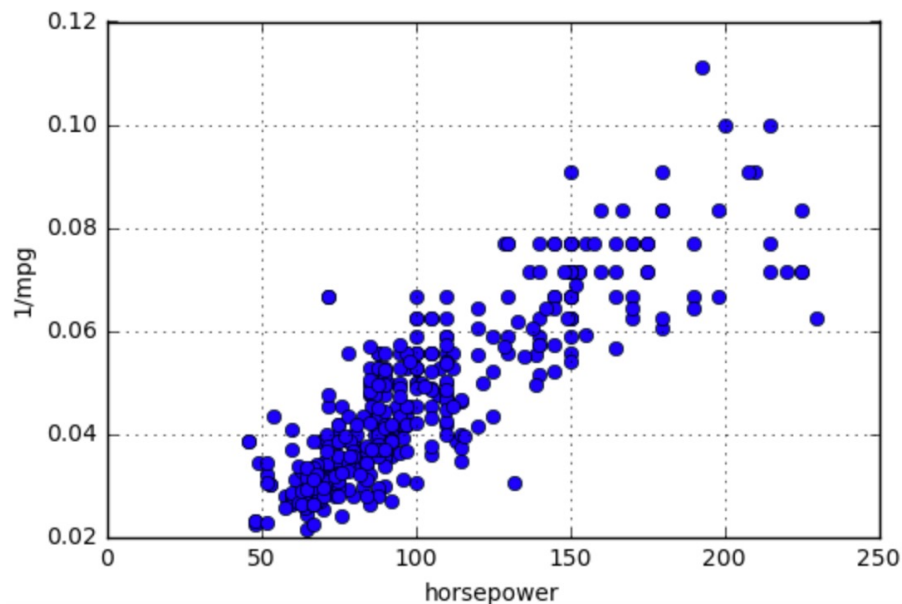
When the Error is Large...



- Many sources of error for a linear model
- Always good to visually inspect the scatter plot
 - Look for trends
- Example to the left
 - All four data sets have same regression line
 - But, errors and their reasons are different
- How would you describe these errors?

A Better Model for the Auto Example

- Fit the inverse: $\frac{1}{\text{mpg}} = \beta_0 + \beta_1 \text{horsepower}$
- Uses a nonlinear transformation
- Will cover this idea later



Test MSE = 21.52 (linear)

Test MSE = 17.75 (linear+inversion)

Next Steps

- Run the code associated with this lecture.
 - Either on your local machine or Google Colab
- Don't be intimidated by the content.
 - Follow the basics of the code, no need to memorize anything.