Machine Learning and Financial Applications

Lecture 2

Linear regression

Liu Peng

liupeng@smu.edu.sg

Video tutorials

- Introducing linear regression
  https://youtu.be/n61tkVF6uAU
- Deriving closed-form solution
  https://youtu.be/-H2hFOFjfoE

The Purpose of Regression Analysis

Explanatory modeling

- How marketing spend affects quarterly sales
- How smoker status affects insurance premium
- How education affects income

Predictive modeling

- Predict quarterly sales, given the marketing spend
- Predict insurance premium, given a policyholder’s age, gender, body mass index (BMI), and smoker status
- Predict income, given the education, age, work experience, industry

Simple Linear Regression (SLR)

Linear equation

$y = \beta_0 + \beta_1 x$

- β_0: intercept with the y-axis
- β_1: coefficient for the input variable

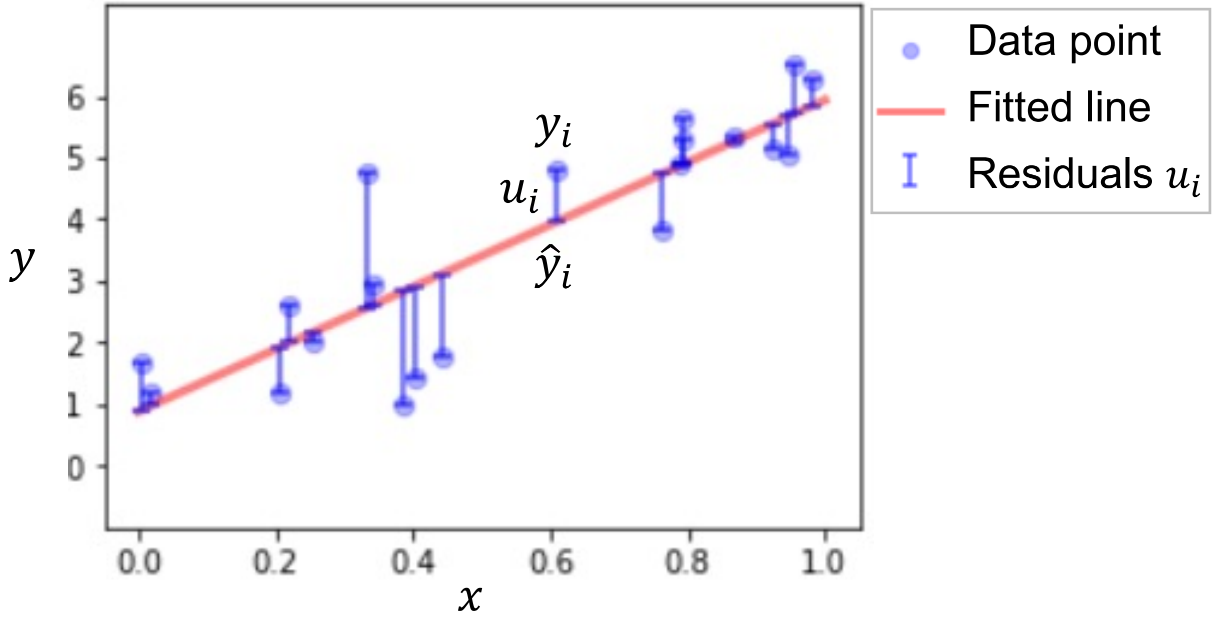
Terminology

y	x
Target variable	**Input variable**
Dependent variable	Independent variable
Explained variable	Explanatory variable
Response variable	Control variable
Predicted variable	Predictor variable
Regressand variable	Regressor

Objective

$$\min \ \mathrm{SSR} = \min \sum_{i=1}^{n} u_i^2 = \min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Minimize the sum of squared residuals (SSR)
- Each residual u_i is the difference between the observation y_i and its fitted value $\hat{y}_i$
- In simple terms, the objective is to find the straight line that is closest to data points given



Scikit-learn: LinearRegression class

- Import LinearRegression **class** (inside sklearn library's linear_model module)
- Prepare x and y for the LinearRegression Model
- Create LinearRegression **object** to fit the x and y observed data points
- Print the results of β_0, β_1 using **object**'s attributes `.intercept_` and `.coef_`

```
from sklearn.linear_model \
import LinearRegression
# Refer to jupyter notebook
file on how to prepare x and y
lm = LinearRegression()
lm.fit(x_orig, y)


print(lm.intercept_)
print(lm.coef_)


x_new = np.array([[100],[200]])
y_pred = lm.predict(x_new)
y_pred
```

What is class and object?

- A **class** is a template with defined attributes and methods; can be considered a data type

- An **object** can be created from the template by calling the class name with brackets

```
lm = LinearRegression()
```

- If we assign the created object to a variable, the variable contains the object with the template's defined attributes and methods

- Use the object's attributes and methods with the dot operator

```
lm.intercept_
lm.coef_
lm.fit(x_orig, y)
lm.predict(x_new)
```

Understanding the results

Mean Squared Error (MSE)

- Mean of the squared differences between observed y and predicted y

Root Mean Squared Error (RMSE)

- Square root of MSE

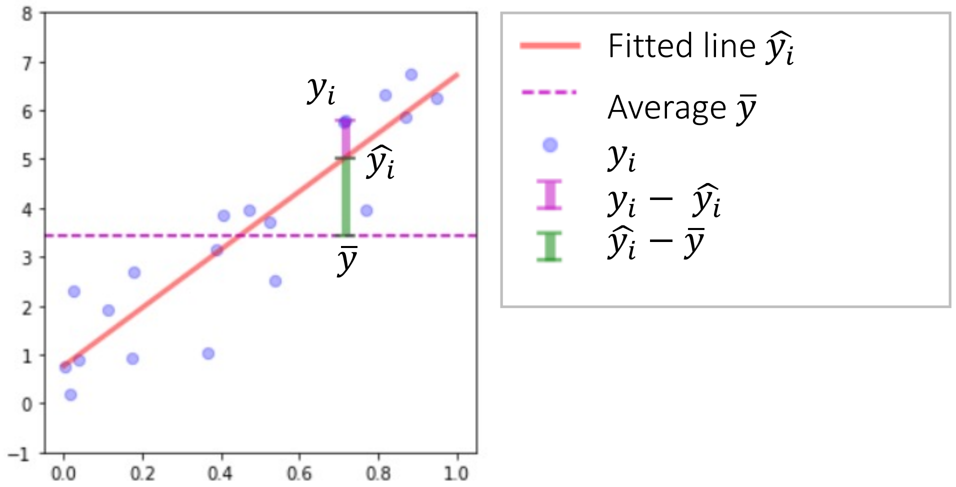
```
from sklearn.metrics import
mean_squared_error
mse = mean_squared_error(y, \
y_pred)
print("MSE:", mse, "RMSE:", \
np.sqrt(mse))
```

R2

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

- Proportion of variance in the observed data that is explained by the model
- The higher the R2, usually the better the linear fit is for the data set (not always)

```
lm.score(x_orig, y)
```



Issue: only one input variable considered in SLR

Multiple Linear Regression (MLR)

Equation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

- β_i: coefficient for i-th input variable

Advantage

- Can accommodate many input variables
- Holistic view of relationship between target and all input variables

Assumptions

- None of the input variables is constant
- **No** perfect linear relationships among the input variables like below

$$x_j = \gamma_0 + \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_k x_k$$

Ceteris paribus analysis

- Ceteris paribus: Latin of "all other things being equal"
- MLR allows us to explicitly control many other factors that simultaneously affect the target variable and observe the impact of only one factor

$$y + \Delta y = \beta_0 + \beta_1 x_1 + \cdots + \beta_j (x_j + \Delta x_j) + \cdots + \beta_k x_k$$

$$\Delta y = \beta_j \, \Delta x_j$$

- E.g., we control all other input variables but only bump x_j, to see the impact on y

MLR implementation for advertising data set

- Prepare x with all input variables, i.e., a DataFrame with multiple columns TV, Newspaper, Radio; prepare y as the sales
- Create an object from LinearRegression class
- Use the regression model to fit x and y and see the model parameters
- R2 statistically usually **increases** every time **an input variable is added**

```
x_all = data.loc[:, 'TV':'Newspaper']
y = data.loc[:,"Sales"]

lm_all_sklearn = LinearRegression()

lm_all_sklearn.fit(x_all, y)
print(lm_all_sklearn.intercept_)
print(lm_all_sklearn.coef_)
lm_all_sklearn.score(x_all,y)
```

Caveat: A regression model with more input variables and higher R2 does **NOT** necessarily mean that the model is a better fit and can predict better

Newspaper vs sales: positive or negative correlation?

Simple linear regression

$$y = \beta_0 + \beta_1 x_{Newspaper}$$

0.0547

Multiple linear regression

$$y = \beta_0 + \beta_1 x_{TV} + \beta_2 x_{Radio} + \beta_3 x_{Newspaper}$$

-0.001

Newspaper spend vs. sales has a positive correlation of 0.228299

```
data.loc[:, ['Newspaper','Sales']].corr()
```

Simple linear regression

- $\beta > 0$ => y and x are positively correlated
- y and x are positively correlated => $\beta > 0$

Multiple linear regression

- $\beta > 0$ ≠> y and x are positively correlated
- y and x are positively correlated ≠> $\beta > 0$

Case study: Car insurance

y : claim amount in one year

x_{age} : age of car

x_{sum} : sum insured (higher sum indicates higher market value)

y vs. x_{age} : **negatively** correlated, because newer cars usually have higher market value; hence higher claim amount

y vs. x_{sum} : positively correlated, i.e., more expensive cars have higher claim amount

$$\text{MLR: } y = \beta_0 + \beta_1 x_{age} + \beta_2 x_{sum}$$

- β_1 is positive, i.e., for a fixed x_{sum}, y has a **positive** relationship with x_{age}
- **For a fixed sum insured**, newer cars have lower claim amount than older cars
  - An old car is likely to be an inherently more prestigious brand or model, while a new car with the same sum insured is likely to be a mass-market brand
  - Due to depreciation, an old car has the same low sum insured as a new car
  - Older cars with high-end brand are more likely to have higher claim amount, maybe because the parts are more expensive etc

How do we fit in categorical variables for the data set of condo transaction?

	name	price	unit_price	district_code	segment	type	area	level	remaining_years	date
0	SEASCAPE	4388000	2028	4	CCR	Resale	2164	06 to 10	87.0	Nov-19
1	COMMONWEALTH TOWERS	1300000	1887	3	RCR	Resale	689	16 to 20	93.0	Nov-19
2	THE TRILINQ	1755000	1304	5	OCR	Resale	1346	06 to 10	92.0	Nov-19
3	THE CREST	2085000	2201	3	RCR	Resale	947	01 to 05	92.0	Nov-19
4	THE ANCHORAGE	1848888	1468	3	RCR	Resale	1259	01 to 05	999.0	Nov-19

Solution: one-hot encoding

Categorical variable

$$x_{type} = \begin{cases} 'Resale' \\ 'New\ Sale' \end{cases}$$

Dummy variable

$$d_{Resale} = \begin{cases} 1\ if\ x_{type} = 'Resale' \\ 0\ if\ x_{type} = 'New\ Sale' \end{cases}$$

For student's own exploration

Option 1: Use pandas.get_dummies()

```
type_dummies = pd.get_dummies(data.type)
data = pd.concat([data, type_dummies], axis=1)
```

Option 2: Use scikit-learn OneHotEncoder

Alternative method - Ordinary Least Squares (OLS)

- Statsmodels library has OLS method, which can handle numerical and **categorical variables efficiently**
- Input '$y = x_1 + x_2 + \cdots$' as a string and data set to create a new OLS object, i.e., a new multiple linear regression model
- Call the fit() method to run the MLR and show the results

```
import statsmodels.formula.api as smf
d5_condo =
data.loc[(data['district_code']==5) &
(data['area']<1500) &
(data['remaining_years']<100)]
d5_model = smf.ols('price ~ area + type',
data=d5_condo)

result = d5_model.fit()
print(result.summary())
```

Understanding the results

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.824
Model:                            OLS   Adj. R-squared:                  0.824
Method:                 Least Squares   F-statistic:                     3271.
Date:                Tue, 19 Apr 2022   Prob (F-statistic):               0.00
Time:                        13:54:04   Log-Likelihood:                -18424.
No. Observations:                1402   AIC:                         3.685e+04
Df Residuals:                    1399   BIC:                         3.687e+04
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept        1.887e+05   1.17e+04     16.087      0.000    1.66e+05    2.12e+05
type[T.Resale]  -1.614e+05   7465.490    -21.624      0.000   -1.76e+05   -1.47e+05
area             1024.6680     12.803     80.035      0.000     999.553    1049.783
```

β_j

$\beta_j \pm 2 * std\ err$

$d_{Resale} =$ type[T.Resale]

95% confidence interval (CI) for the truth of β_j

$$\begin{cases} H_0: \beta_j = 0 & \textbf{Null hypothesis} \\ H_a: \beta_j \neq 0 & \textbf{Alternative hypothesis} \end{cases}$$

P-value is the probability for H_0 to be true; hence if P-value < significance level (typically 0.05), we prove H_0 false, and hence H_a true

Effectively we have a different linear equation for each value of d_{resale}

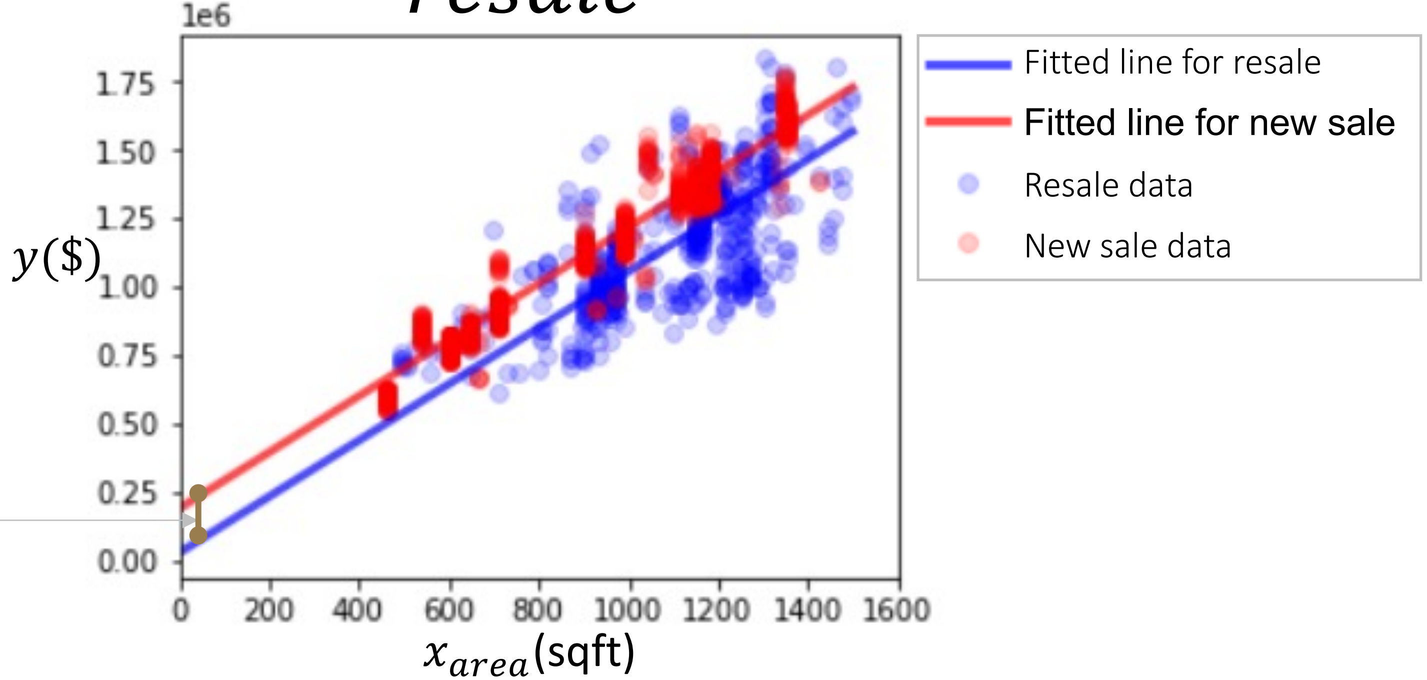
$$y = \beta_0 + \beta_1 d_{resale} + \beta_2 x_{area}$$

Resale condo

$$d_{Resale} = 1 \;\; => y = \beta_0 + \boxed{\beta_1} + \beta_2 x_{area}$$

New sale condo

$$d_{Resale} = 0 => y = \beta_0 + \beta_2 x_{area}$$



district_code	segment	type	area	level
4	CCR	Resale	2164	06 to 10
3	RCR	Resale	689	16 to 20
5	OCR	Resale	1346	06 to 10
3	RCR	Resale	947	01 to 05
3	RCR	Resale	1259	01 to 05

Question: how to deal with categorical variables with more than 2 values?

How OLS deals with categorical variable with multiple values

	d_B	…	d_Z
A	0		0
B	1		0
…	…		
Z	0		1

- One value (e.g., A) is set as baseline value
- All other values become a binary variable
- n values means n-1 dummy variables

Question: Why not n dummy variables? Why do we exclude the baseline value?

- If we have d_A, what is the relationship between d_A, d_B, … d_Z?
- Hence it adds no value to include d_A

A different linear equation for each value of segment variable

$$y = \beta_0 + \beta_1 d_{OCR} + \beta_2 d_{RCR} + \beta_3 x_{area}$$

```
small_condo = data.loc[data['area'] < 1500]
small_condo_model = smf.ols('price ~ segment + area', data=small_condo)
result = small_condo_model.fit()
print(result.summary())
```

CCR condo

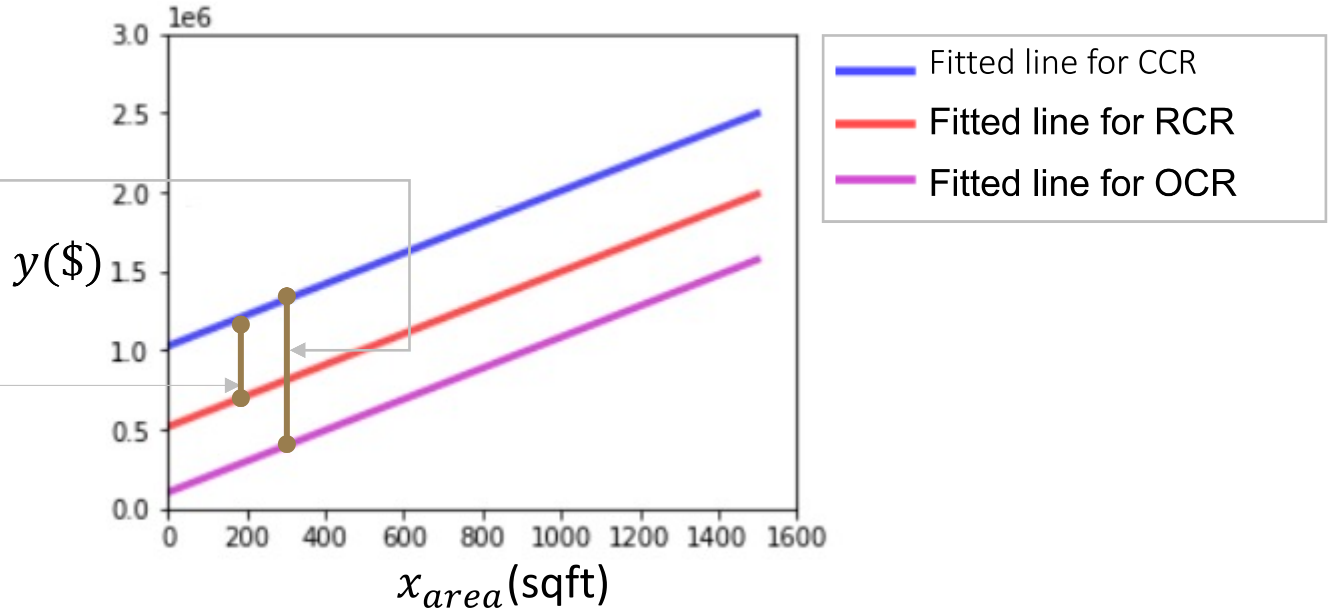
$$y = \beta_0 + \beta_3 x_{area}$$

OCR condo

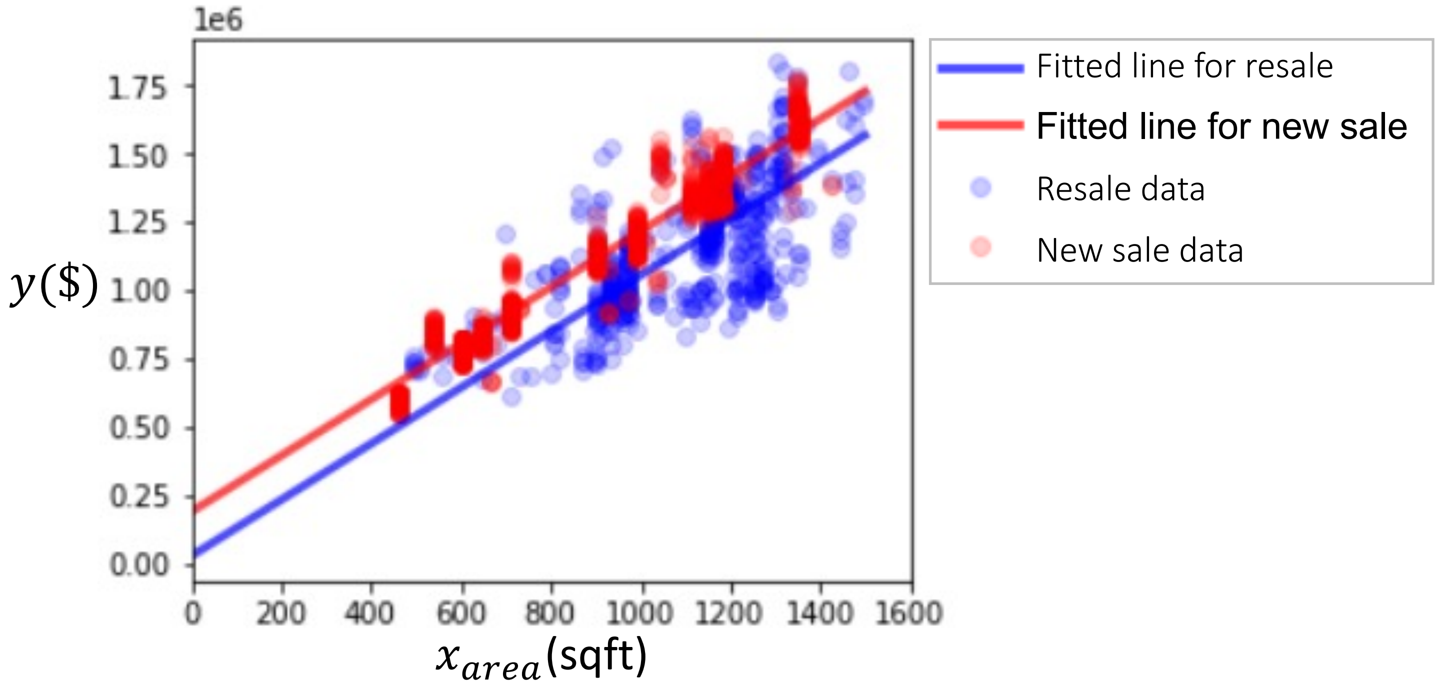
$$y = \beta_0 + \boxed{\beta_1} + \beta_3 x_{area}$$

RCR condo

$$y = \beta_0 + \boxed{\beta_2} + \beta_3 x_{area}$$



Resale vs. New Sale: should the gradient of the two linear equations be the same?



Issue: Should resale condo's price per sqft be the same as new sale condo?

Solution: Interaction terms

Interaction terms

$$y = \beta_0 + \beta_1 d_{resale} + \beta_2 x_{area} + \beta_3 d_{resale} \cdot x_{area}$$

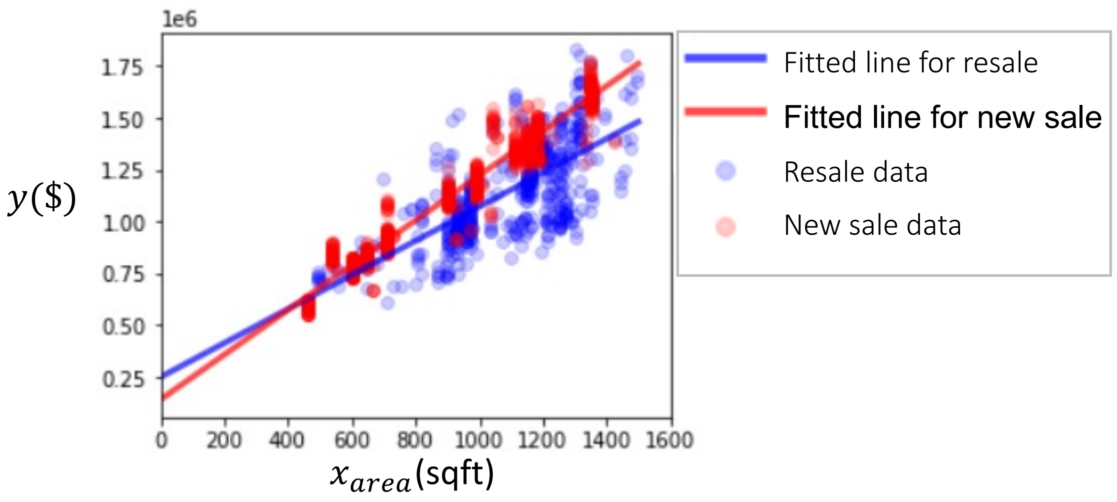
```
d5_model = smf.ols('price ~ type * area', d5_condo)
result = d5_model.fit()
print(result.summary())
```

Resale condo

$$d_{Resale} = 1 \;\; => y = \beta_0 + \beta_1 + (\beta_2 + \beta_3) x_{area}$$

New sale condo

$$d_{Resale} = 0 => y = \beta_0 + \beta_2 x_{area}$$



Did the interaction term improve the model?

$$y = \beta_0 + \beta_1 d_{resale} + \beta_2 x_{area}$$

OLS Regression Results

Dep. Variable:	price	R-squared:	0.824
Model:	OLS	Adj. R-squared:	0.824
Method:	Least Squares	F-statistic:	3271.
Date:	Tue, 19 Apr 2022	Prob (F-statistic):	0.00
Time:	13:54:04	Log-Likelihood:	-18424.
No. Observations:	1402	AIC:	3.685e+04
Df Residuals:	1399	BIC:	3.687e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.887e+05	1.17e+04	16.087	0.000	1.66e+05	2.12e+05
type[T.Resale]	-1.614e+05	7465.490	-21.624	0.000	-1.76e+05	-1.47e+05
area	1024.6680	12.803	80.035	0.000	999.553	1049.783

$$y = \beta_0 + \beta_1 d_{resale} + \beta_2 x_{area} + \beta_3 d_{resale} \cdot x_{area}$$

OLS Regression Results

Dep. Variable:	price	R-squared:	0.833
Model:	OLS	Adj. R-squared:	0.832
Method:	Least Squares	F-statistic:	2317.
Date:	Tue, 19 Apr 2022	Prob (F-statistic):	0.00
Time:	13:26:27	Log-Likelihood:	-18388.
No. Observations:	1402	AIC:	3.678e+04
Df Residuals:	1398	BIC:	3.681e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef		std err	t	P>|t|	[0.025	0.975]
Intercept	1.408e+05		1.27e+04	11.051	0.000	1.16e+05	1.66e+05
type[T.Resale]	1.081e+05	β_1	.24e+04	3.333	0.001	4.45e+04	1.72e+05
area	1080.5225	β_2	14.100	76.631	0.000	1052.862	1108.183
type[T.Resale]:area	-258.8491	β_3	30.355	-8.527	0.000	-318.395	-199.304

When do we use interaction terms?

Most commonly, to indicate that the relationship between y and a continuous x might be different for subgroups (indicated by categorical variable)

- Female vs. Male: the relationship between weight (y) and height (x) might be different
- Degree holder vs. non-holder: the relationship between salary (y) and years of experience (x) might be different

How to treat numbers as categorical variables

district_code
4
3
5
3
3

- **Issue**: district_code is read into the DataFrame as an integer column. But it has no numerical meaning and should be a categorical variable
- **Solution**: put a C() around the column name

```python
small_condo_model = smf.ols('price ~ district_code + area', data=small_condo)
result = small_condo_model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.448
Model:                            OLS   Adj. R-squared:                  0.448
Method:                 Least Squares   F-statistic:                 1.075e+04
Date:                Tue, 19 Apr 2022   Prob (F-statistic):               0.00
Time:                        14:15:05   Log-Likelihood:            -3.7747e+05
No. Observations:               26480   AIC:                         7.550e+05
Df Residuals:                   26477   BIC:                         7.550e+05
Df Model:                           2
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept      7.477e+05   9347.102     79.991      0.000    7.29e+05    7.66e+05
district_code -2.798e+04    338.519    -82.662      0.000   -2.86e+04   -2.73e+04
area            999.3194      7.989    125.080      0.000     983.660    1014.979
```

```python
small_condo_model = smf.ols('price ~ C(district_code) + area', data=small_condo)
result = small_condo_model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.705
Model:                            OLS   Adj. R-squared:                  0.705
Method:                 Least Squares   F-statistic:                     2526.
Date:                Tue, 19 Apr 2022   Prob (F-statistic):               0.00
Time:                        14:15:43   Log-Likelihood:            -3.6919e+05
No. Observations:               26480   AIC:                         7.384e+05
Df Residuals:                   26454   BIC:                         7.386e+05
Df Model:                          25
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Intercept               9.771e+05    4.1e+04     23.832      0.000    8.97e+05    1.06e+06
C(district_code)[T.2]  -2.772e+05   6.17e+04     -4.496      0.000   -3.98e+05   -1.56e+05
C(district_code)[T.3]  -3.864e+05    4.1e+04     -9.429      0.000   -4.67e+05   -3.06e+05
```

OLS can handle nonlinear terms

- The Meaning of "Linear" in Regression Analysis is that the equation is linear in the parameters $\beta_0, \beta_1, \beta_2$ …

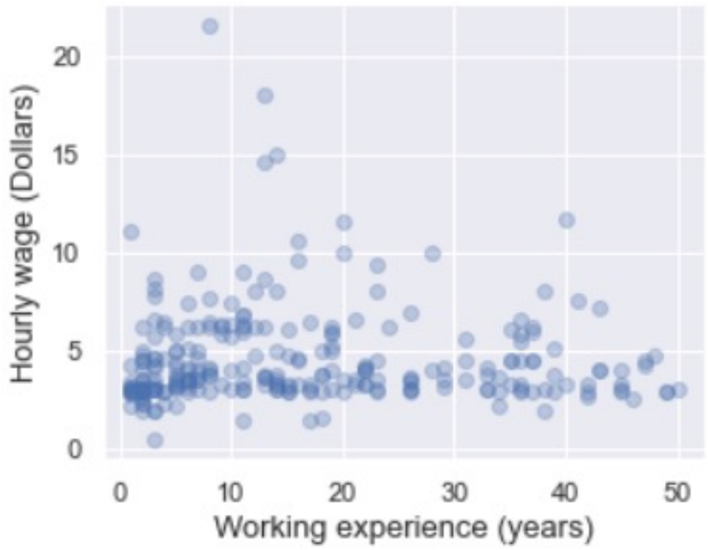
- Below can all be viewed as linear regression models

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2$$
$$y = \beta_0 + \beta_1 \log(x_1)$$
$$\sqrt{y} = \beta_0 + \beta_1 x_1 + \beta_2 \log(x_1)$$

- Below **canNOT** b $y = \beta_0 + \beta_1^x$ s linear regression model

Nonlinear terms in wage data set (1/2)

- Wage data set: wages of several working individuals in 1976
- Can the relationship be fitted to a straight line based on the scatter plot of only female workers?



- Try the two models and interpret which one might be better

Model 1:
$$y_{wage} = \beta_0 + \beta_1 x_{exper}$$

Model 2:
$$y_{wage} = \beta_0 + \beta_1 x_{exper} + \beta_2 \sqrt{x_{exper}}$$

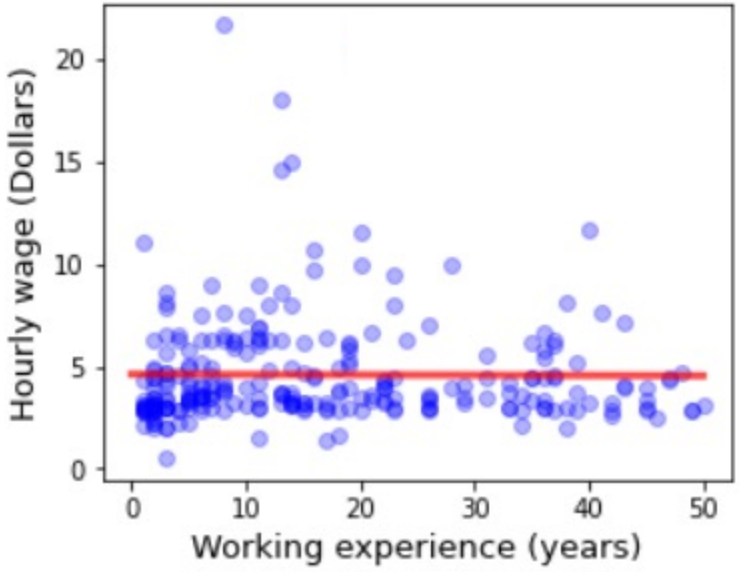
```
model1 = smf.ols('wage ~ exper', data=wage_female)
result1 = model1.fit()
print(result1.summary())


model2 = smf.ols('wage ~ exper + np.sqrt(exper)',
data=wage_female)
result2 = model2.fit()
print(result2.summary())
```

Nonlinear terms in wage data set (2/2)

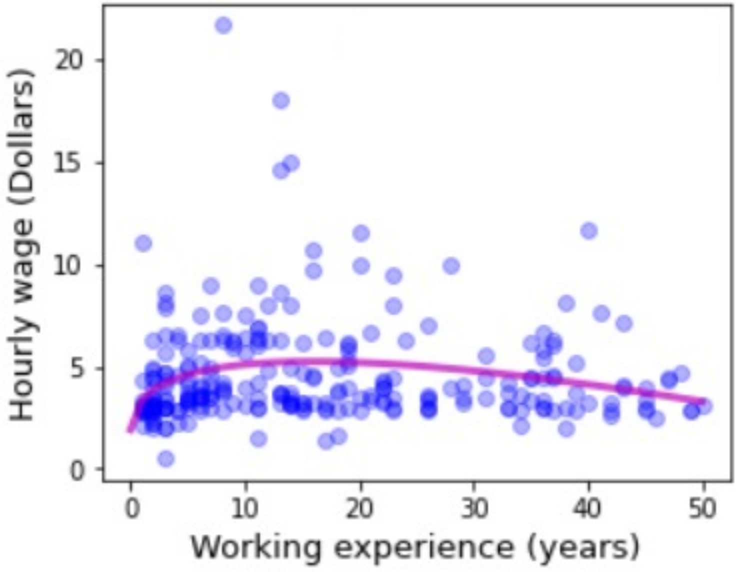
Model 3 $\log(y_{wage}) = \beta_0 + \beta_1 x_{exper} + \beta_2 \sqrt{x_{exper}}$

```
model3 = smf.ols('np.log(wage) ~ exper + np.sqrt(exper)', data=wage_female)
result3 = model3.fit()
print(result3.summary())
```

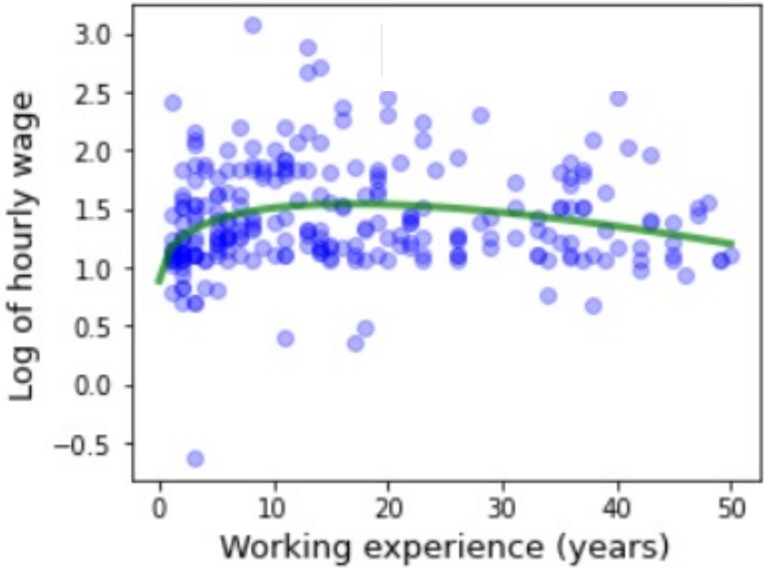
Model 1



Model 2

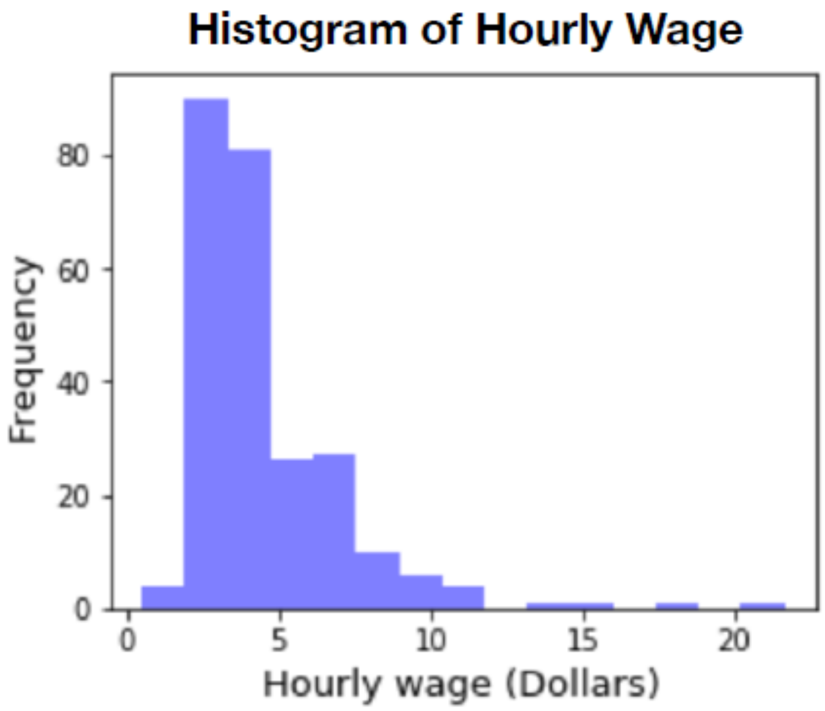


Model 3

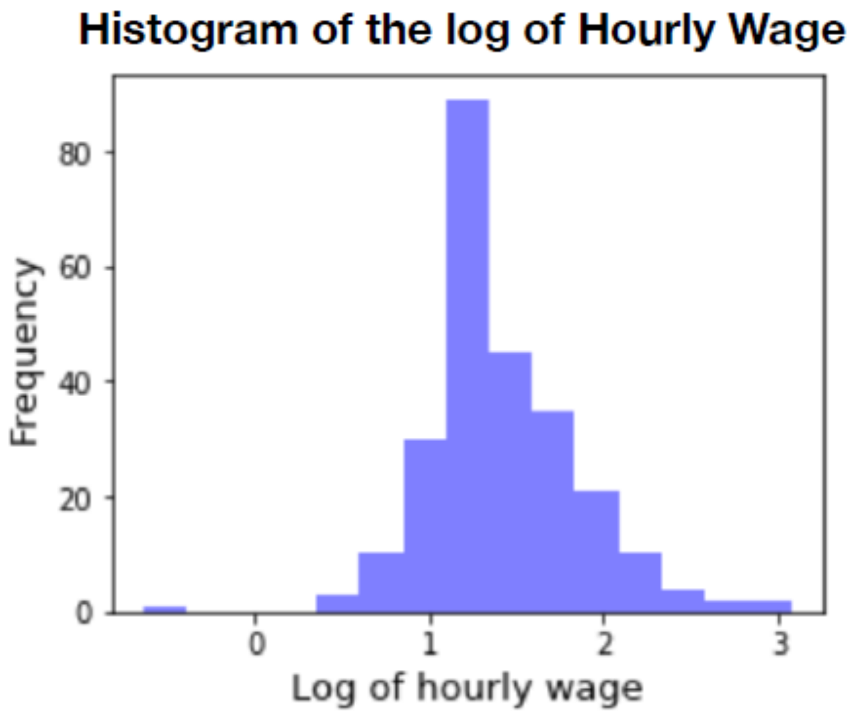


Why can the log term improve the model?

Model 3 $\log(y_{wage}) = \beta_0 + \beta_1 x_{exper} + \beta_2 \sqrt{x_{exper}}$



- Outliers with high wage affect the model performance



- Log(wage) reduces the impact of outliers
- Distribution is closer to a bell curve

Log() guaranteed to work for variables with outliers?

Deriving the closed-form solution

Coding exercise

- Data set: insurance.csv
- Columns: age, sex, bmi, children, smoker, region, charges
- Questions for exploration:
  - Which variable is the y?
  - Which variables are categorical? Which ones are numerical?
  - Any interaction terms?
  - Any non-linear terms?

Recap and next lecture

Recap

- Linear Regression
- Scikit-learn LinearRegression Class
- Statsmodels OLS method
- Simple Linear Regression
- Multiple Linear Regression
- Coefficient vs. correlation
- Categorical variables with 2 or more values
- Interaction terms
- Treat numbers as categorical variables
- Nonlinear terms

Next lecture

- Logistic Regression

Homework

- Review video tutorials, class materials and recordings to week 2
- Go through the accompanying coding material (to be uploaded after week 2 class)
- Post learning reflections and questions if any
- Complete group assignment and submit as a group before week 3 lecture