Descriptive Statistics

Churn_Modelling Dataset

: ชุดข้อมูล ดูอัตราร้อยละต่อปีที่ลูกค้าหยุดสมัครรับบริการหรือพนักงานออกจากงาน

CODE part

```
from google.colab import files
uploaded = files.upload()
```

```
import pandas as pd
df = pd.read_csv( 'Churn_Modelling.csv' )
```

Upload file + ใช้ pandas read .csv

returns a “ pandas dataframe “

```
df.dtypes
```

บอก data type แต่ละ column ใน dataframe

Descriptive Statistics for Numeric Data

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

// ไม่มี mode

Week 1

Mode (.mode())

```
df.mode()
df.mode(numeric_only=True)
df["Gender"].mode()
```

//ฐานนิยาม

Variance

```
df.var()
df.var()['Age']
```

Coefficient of Variation

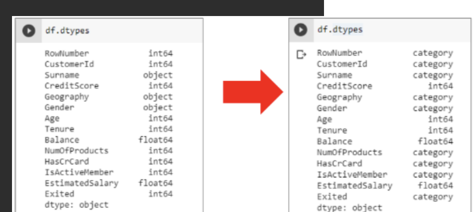
```
from scipy.stats import variation
variation(df.var['Age'])
```

Descriptive Statistics for Categorical Data

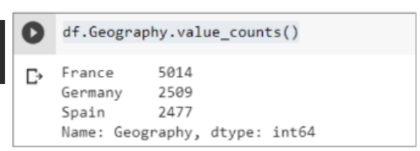
```
df.describe(exclude=['float', 'int64'])
df.describe(include = 'object')
# หลัง convert data type ไปใช้ 'category'
```

Convert Data Type (.astype ())

```
df.RowNumber=df.RowNumber.astype('category')
df.CustomerId=df.CustomerId.astype('category')
df.HasCrCard=df.HasCrCard.astype('category')
df.IsActiveMember=df.IsActiveMember.astype('category')
df.Exited=df.Exited.astype('category')
df.NumOfProducts=df.NumOfProducts.astype('category')
df.Geography = df.Geography.astype('category')
df.Surname = df.Surname.astype('category')
df.Gender = df.Gender.astype('category')
```

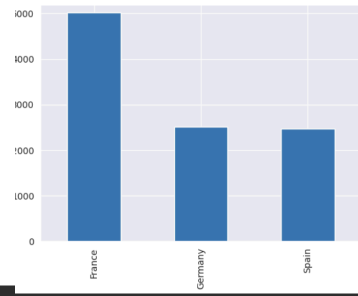



```
df.Geography.value_counts()
```



Week 1

“ Data Visualization “



matplotlib : สร้างกราฟใน Python

seaborn : ใช้ปรับ style ของกราฟเพื่อความสวยงาม

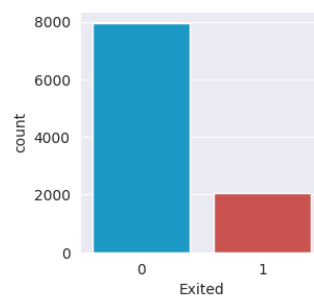
Bar charts

```
df.Geography.value_counts().plot.bar(grid=False)
```

// pandas ทำได้

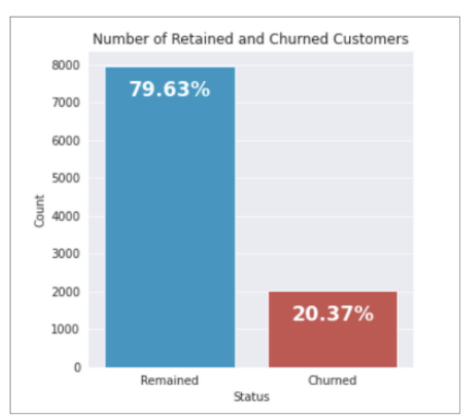
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid') #{darkgrid, whitegrid, dark, white, ticks}
colors = ['#00A5E0', '#DD403A']
```

```
fig = plt.figure(figsize = (5, 5))
sns.countplot(x = 'Exited', data = df, palette = colors)
```




```
for index, value in enumerate(df['Exited'].value_counts()):
  # label = '{}%'.format(round((value/df['Exited'].shape[0])*100, 2))
  #เป็น %
  label = '{:,}'.format(value)
  #เป็นจำนวณ
  plt.annotate(label, xy = (index -0.25, value -1000), color =
'w',fontweight='bold',size=17) #จัดตำแหน่งตัว % ในกราฟ
```

```
plt.title('Number of Retained and Churned Customers')
plt.xticks([0, 1], ['Remained', 'Churned'])
#ในไฟล์ที่เป็น 0,1 ใน column ['Exited']
plt.xlabel('Status')
plt.ylabel('Count');
```

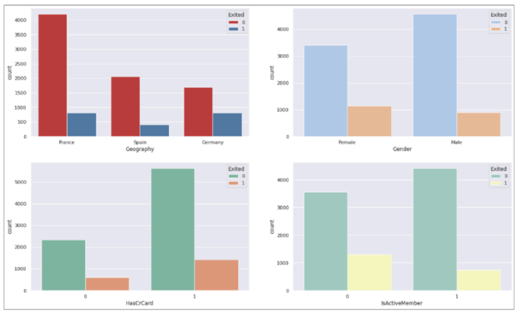


Week 1

```
fig, axarr = plt.subplots(2, 2, figsize=(20, 12))

sns.countplot(x='Geography', hue = 'Exited',palette="Set1",data = df, ax=axarr[0][0])
sns.countplot(x='Gender', hue = 'Exited',data = df, ax=axarr[0][1])
sns.countplot(x='HasCrCard', hue = 'Exited',data = df, ax=axarr[1][0])
sns.countplot(x='IsActiveMember', hue = 'Exited',data = df, ax=axarr[1][1])
```

Palette = {(deep, muted, bright, pastel, dark, colorblind)}

```
sns.countplot(x='Geography', hue='Exited',data = df, palette="Set1", ax=axarr[0][0])
sns.countplot(x='Gender', hue ='Exited',data = df, palette="pastel", ax=axarr[0][1])
sns.countplot(x='HasCrCard', hue='Exited',data = df, palette="Set2", ax=axarr[1][0])
sns.countplot(x='IsActiveMember', hue='Exited',data = df, palette="Set3", ax=axarr[1][1])
```



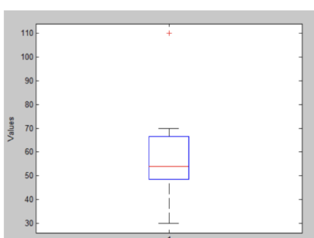
19

Box Plot

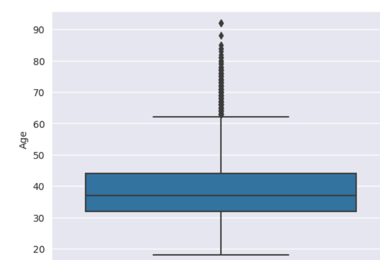
Box Plot

```
sns.boxplot(y='Age',data = df)
```

Example: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110



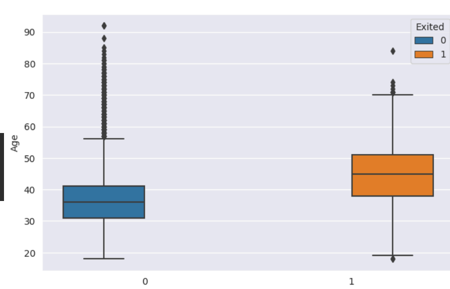
$Q_1 = 48.5$
$Q_2 = 54$
$Q_3 = 66.5$
$IQR = 18$

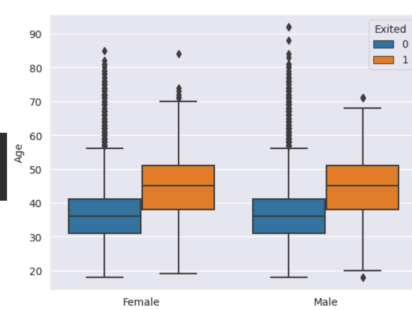



```
fig = plt.figure(figsize = (8, 5))
sns.boxplot(y='Age',x = 'Exited', hue = 'Exited',data = df)
```

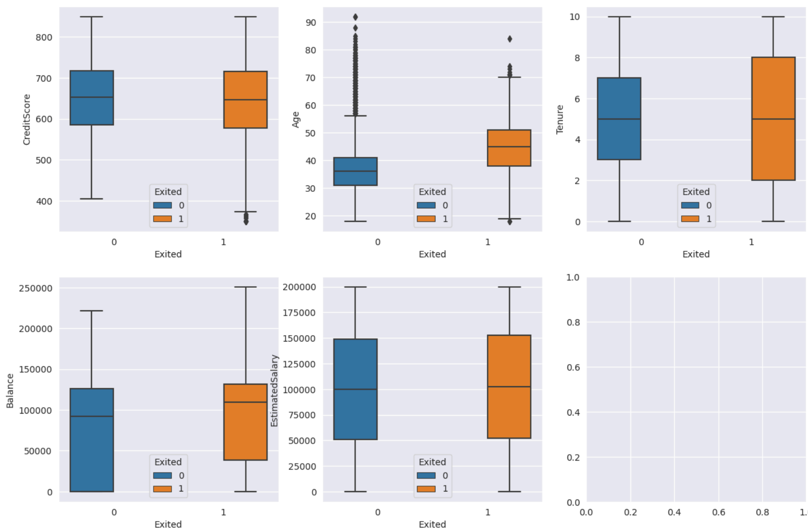
```
sns.boxplot(y='Age',x= 'Gender', hue='Exited',data =df)
```

```
fig,                figsize=(15,
```



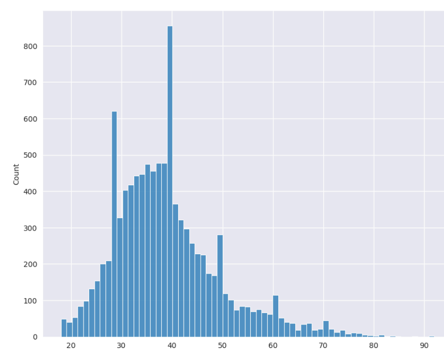


```
fig, axarr = plt.subplots( 2, 3, figsize=(15, 10))

sns.boxplot(y='CreditScore',x = 'Exited', hue = 'Exited',data = df, ax=axarr[0][0])
sns.boxplot(y='Age',x = 'Exited', hue = 'Exited',data = df , ax=axarr[0][1])
sns.boxplot(y='Tenure',x = 'Exited', hue = 'Exited',data = df, ax=axarr[0][2])
sns.boxplot(y='Balance',x = 'Exited', hue = 'Exited',data = df, ax=axarr[1][0])
sns.boxplot(y='EstimatedSalary',x = 'Exited', hue = 'Exited',data = df, ax=axarr[1][1])
```

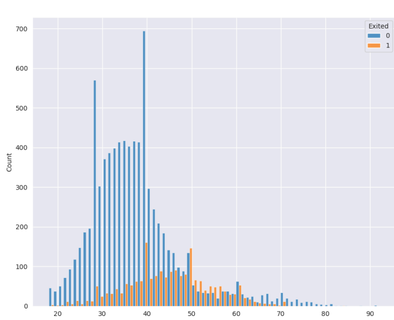


Histogram

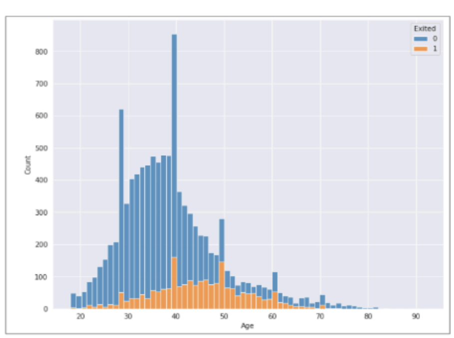
```
fig = plt.figure(figsize = (10,8))
sns.histplot(df, x="Age")
```



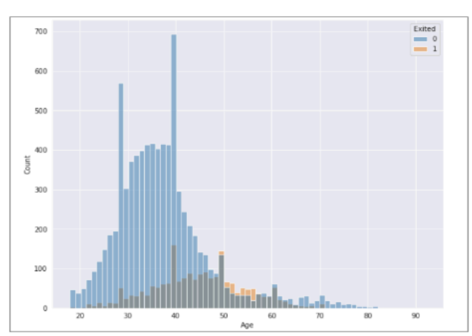

```
fig = plt.figure(figsize = (10, 8))
sns.histplot(df, x="Age", hue = 'Exited',multiple="dodge")
```



multiple="stack"

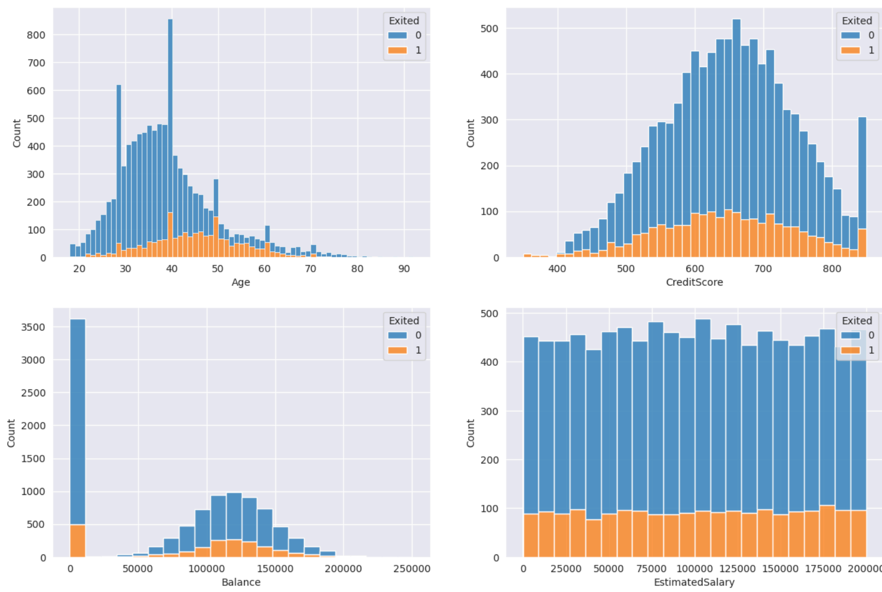


multiple="layer"



Week 1

```
fig, axarr = plt.subplots( 2, 2, figsize=(15, 10))
sns.histplot(df, x="Age", hue = 'Exited',multiple="stack", ax=axarr[0][0])
sns.histplot(df, x="CreditScore", hue = 'Exited',multiple="stack", ax=axarr[0][1])
sns.histplot(df, x="Balance", hue = 'Exited',multiple="stack", ax=axarr[1][0])
sns.histplot(df, x="EstimatedSalary", hue = 'Exited',multiple="stack", ax=axarr[1][1])
```



Week 2

Hypothesis Testing

Z-test

//skip ไปอ่านจากตัวอย่างก็ได้

: Test for p1-p2

320 of 400 people asked in North //p1

300 of 425 people asked in South //p2

- Ho: p1 -p2 = 0
- Ha: p1 - p2 $\neq$ 0
- two-tailed test with α = 0.05

```
from statsmodels.stats.proportion import proportions_ztest
import numpy as np
```

```
significance = 0.05 #alpha value
successes = np.array([320, 300])
samples = np.array([400, 425])
```

Compute z-statistics and p-value

```
stat,p_value =
proportions_ztest(count=successes,nobs=samples,alternative='two-sided')
# alternative : str in ['two-side','smaller','lager']
# lager : p1 > p2
# smaller : p1 < p2
```

stat = Z

'two-sided' เพราะ เป็น two tail

```
print('z_stat: %0.5f, p_value: %0.5f' % (stat, p_value))
if p_value < significance:
  print ("Reject the null hypothesis")
else:
  print ("Accept the null hypothesis")
```

```
z_stat: 3.12644, p_value: 0.00177
Reject the null hypothesis
```

Week 2

ตัวอย่าง Z-test

Z-Test for the Difference in Two Proportions: **Heart Disease dataset** Use z-test for p1-p2

Ho: p1 -p2 = 0 // H0 = gender ไม่มีผล p1=p2

Ha: p1 - p2 ≠ 0

two-tailed test with $\alpha = 0.01$

- **p1** is the proportion of **females** <u>having heart disease</u>
- **p2** is the proportion of **males** <u>having heart disease</u>

// upload file + import numpy and pandas ก่อน//

```
df = pd.read_csv('HeartDisease.csv')
df['Gender'] = df.sex.replace({1: 'Male', 0: 'Female'})
```

```
p = df.groupby('Gender')['target'].agg([lambda z: np.sum(z==1),'size'])
# np.sum(z==1) : นับจำนวนแถวที่เป็น 1 # ['target'] == 1 : have heart disease
# groupby เพศ : female/male
p.columns = ['HeartDisease', 'Total']
# total คือ จำนวนข้อมูลทั้งหมด ทั้ง0,1
```

compute Z

```
from statsmodels.stats.proportion import proportions_ztest
significance = 0.01
successes = np.array([ p.HeartDisease.Female, p.HeartDisease.Male ])
samples = np.array([ p.Total.Female, p.Total.Male ])
stat, p_value = proportions_ztest(count=successes, nobs=samples,
alternative='two-sided')
```

```
print('z_stat: %0.5f, p_value: %0.6f' % (stat, p_value))
if p_value < significance:
  print ("Reject the null hypothesis")
else:
  print ("Accept the null hypothesis")

z_stat: 4.89023, p_value: 0.000001
Reject the null hypothesis
```

หมายความว่า Reject H0 ⇒ p1 != p2 ⇒ เพศมีผลต่อ heart diseart

t-test on different mean(x-y)

Test on different means µx-µy

```
import numpy as np
import scipy.stats as stats
```

```
#equal_var = True
significance = 0.05
A = np.array([43, 53, 65, 49, 55, 60, 47, 50, 60, 55])
B = np.array([62, 43, 54, 67, 59, 45, 46, 63, 65, 45])

#equal_var = False
significance = 0.05
A = np.array([43, 53, 65, 49, 55, 60, 147, 50, 60, 55])
B = np.array([62, 43, 54, 67, 59, 45, 46, 63, 65, 45])
```

****compute T****

```
stat, p_value = stats.ttest_ind(A,B, equal_var = True)
```

equal_var bool, optional

If True (default), perform a standard independent 2 sample test that assumes equal population variances [1]. If False, perform Welch’s t-test, which does not assume equal population variance [2].

กรณีที่

- $\sigma_x^2 \neq \sigma_y^2$ (unknown)

เปลี่ยน equal_var = False

```
print('t_stat: %0.5f, p_value: %0.4f' % (stat, p_value))
if p_value < significance:
 print ("Reject the null hypothesis")
else:
 print ("Accept the null hypothesis")

t_stat: -0.32795, p_value: 0.7467
Accept the null hypothesis
```

Week 2

Paired t-test

Paired t-Test (two samples are dependent)

```
significance = 0.01
group1 = np.array([60, 45, 80, 87, 79, 75, 60, 30, 45])
group2 = np.array([75, 65, 90, 80, 89, 95, 85, 69, 40])
```

```
stat, p_value = stats.ttest_rel(group1,group2)
```

```
print('t_stat: %0.5f, p_value: %0.4f' % (stat, p_value))
if p_value < significance:
 print ("Reject the null hypothesis")
else:
 print ("Accept the null hypothesis")

t_stat: -2.94514, p_value: 0.0186
Accept the null hypothesis
```

ตัวอย่าง Paired t-Test: Blood Pressure Difference

```
import numpy as np
import pandas as pd
import scipy.stats as stats
```

```
df = pd.read_csv('BloodPressure.csv')
df[['bp_before','bp_after']].describe()
```

	bp_before	bp_after
count	120.000000	120.000000
mean	156.450000	151.358333
std	11.389845	14.177622
min	138.000000	125.000000
25%	147.000000	140.750000
50%	154.500000	149.500000
75%	164.000000	161.000000
max	185.000000	185.000000

The blood pressure before the treatment was higher (156.45 ± 11.39) compared to the blood pressure after treatment (151.36 ± 14.18)

```
significance = 0.01
stat,p_value = stats.ttest_rel(df['bp_before'], df['bp_after'])
```

```
print('t_stat: %0.5f, p_value: %0.4f' % (stat, p_value))
if p_value < significance:
  print ("Reject the null hypothesis")
else:
  print ("Accept the null hypothesis")

t_stat: 3.33719, p_value: 0.0011
Reject the null hypothesis
```

There is a statistically significant decrease(ลดลง) in blood pressure

---

Chi-square test

```
from scipy.stats import chi2_contingency
df = pd.DataFrame(index=["Married","Single"], data={'Male': [25,
35],'Female': [15,25]})
```

	Male	Female
Married	25	15
Single	35	25

```
chi2, p, dof, expected = chi2_contingency(df,correction=False)
print(f"chi2 statistic: {chi2:.5g}")
print(f"p-value: {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:")
print(expected)

chi2 statistic: 0.17361
p-value: 0.67692
degrees of freedom: 1
expected frequencies:
[[24. 16.]
 [36. 24.]]
```

Week 2

ตัวอย่าง Chi-square test

If there is a relationship between **sex** and **heart disease** at =1%

//Dataset เดียวกับ ตัวอย่าง Z test

```
df['target'].replace({1:'Yes', 0:'No'},inplace=True)
Table1 = pd.crosstab(df.Gender, df.target, margins=True) #จะมี column All
Table1 = pd.crosstab(df.Gender, df.target)
# crosstab  : count ให้เลย >> เราเลือกแถว คอมลัม
```

```
from scipy.stats import chi2_contingency
chi2, p, dof, expected = chi2_contingency(Table1,correction=False)
print(f"chi2 statistic: {chi2:.5g}")
print(f"p-value: {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:")
print(expected)
significance = 0.01
if p < significance:
  print ("sex and have heart disease are dependent")
else:
  print ("sex and have heart disease are independent")

chi2 statistic: 23.914
p-value: 1.0072e-06
degrees of freedom: 1
expected frequencies:
[[ 43.72277228  52.27722772]
 [ 94.27722772 112.72277228]]
sex and have heart disease are dependent
```

Regression

Simple linear regression

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
```

```
x = np.array([50, 51, 52, 53, 54]).reshape(-1,1)
y = np.array([20, 40, 50, 70, 80]).reshape(-1,1)
```

```
model=LinearRegression()
model.fit(x,y)
```

```
print('intercept:', model.intercept_)
print('slope:', model.coef_)

intercept: [-728.]
slope: [[15.]]
# Y = -728. + 15x
```

```
#Predict new input x = 51.5
y_predict=model.predict([[51.5]])
y_predict

array([[44.5]])
```

Week3

Case study on linear regression

```
df= pd.read_csv( 'advertising.csv' )
x_TV=df.TV.values.reshape(-1,1)
y=df.Sales.values.reshape(-1,1)
```

Create simple linear regression model

```
model=LinearRegression()
model.fit(x_TV,y)
```

View the model

```
model.intercept_, model.coef_

(array([7.03259355]), array([[0.04753664]]))
```

$\widehat{Y}$ = 7.0326 + (0.0475 * TV)

predict

```
newX=[[300],[500],[1000]]
y_predict=model.predict(newX)
y_predict

array([[21.29358568],
       [30.80091377],
       [54.56923398]])
```

Model Evaluation

R^2 : Coefficient of determination

```
model.score(x_TV,y)

0.611875050850071
```

Mean Absolute Error (MAE)

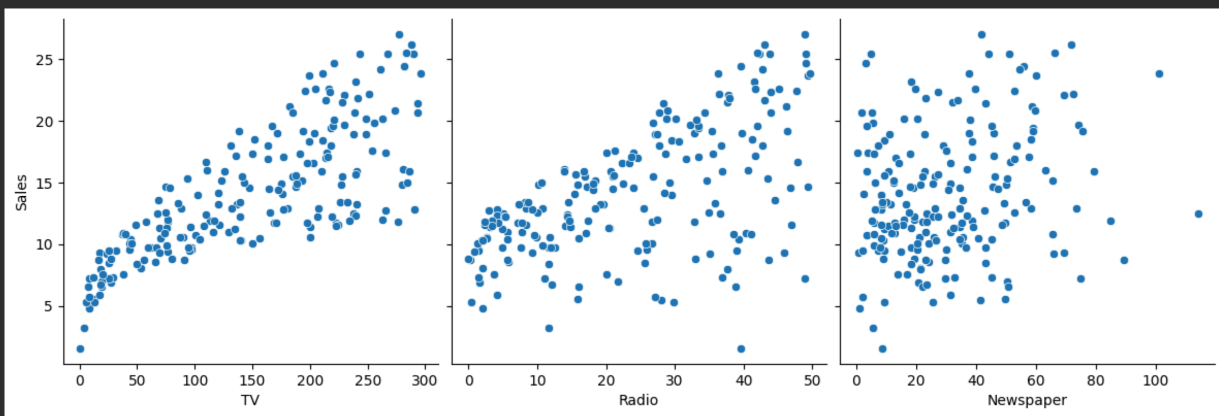
Mean Square Error (MSE)

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

y_predict1=model.predict(x_TV)
```

```
print('MAE =', mean_absolute_error(y,y_predict1))
print('MSE =', mean_squared_error(y,y_predict1))

MAE = 2.549806038927486
MSE = 10.512652915656757
```

Use multiple linear regression

```
import seaborn as sns
sns.pairplot(df, x_vars=['TV','Radio','Newspaper'], y_vars='Sales',
height=4)
```



step1

```
x_Radio=df.Radio.values.reshape(-1,1)
x_News=df.Newspaper.values.reshape(-1,1)
```

```
model.fit(x_Radio,y)
#print(model.score(x_Radio,y))
model.fit(x_News,y)
#print(model.score(x_News,y))
```

step2

```
x_TVRadio=df[['TV','Radio']]
x_TVNews=df[['TV','Newspaper']]
```

Week3

```
model.fit(x_TVRadio,y)
#print(model.score(x_TVRadio,y))
model.fit(x_TVNews,y)
#print(model.score(x_TVNews,y))
```

step3

```
X3=df[['TV','Radio','Newspaper']]
model.fit(X3,y)
#print(model.score(X3,y))
```

```
print(model.coef_)
print(model.intercept_)

[[ 0.04576465  0.18853002 -0.00103749]]
[2.93888937]
```

$\widehat{Y}$ = 2.9389 + (0.04585 * TV) + (0.1885 * Radio) + (-0.0010 * Newspaper)

Prediction

```
x_input=[[300,0,0],[0,300,0],[0,0,300],[100,200,200],[100,200,0]]
model.predict(x_input)

array([[16.66828301], //case1
       [59.49789444], //case2
       [ 2.62764146], //case3
       [45.01385869], //case4
       [45.2213573 ]]) //case5

Case 2 4 5 คือ predict ว่าจะได้ยอดขายดี
```

Week3

Polynomial regression

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

```
x = np.array([0, 1, 2, -1, -2]).reshape(-1,1)
y = np.array([1, 6, 17, 2, 9]).reshape(-1,1)
poly_features = PolynomialFeatures(degree=2)
x_poly=poly_features.fit_transform(x)
model=LinearRegression()
model.fit(x_poly,y)

print('intercept:', model.intercept_)
print('slope:', model.coef_)
```

```
intercept: [1.]
slope: [[0. 2. 3.]]
```

$\widehat{Y} = 1 + 2x^1 + 3x^2$

Case study on polynomial regression

salary.csv

	A	B	C
1	Position	Level	Salary
2	Business Analyst	1	45,000.00
3	Junior Consultant	2	50,000.00
4	Senior Consultant	3	60,000.00
5	Manager	4	80,000.00
6	Country Manager	5	110,000.00
7	Region Manager	6	150,000.00
8	Partner	7	200,000.00
9	Senior Partner	8	300,000.00
10	C-level	9	500,000.00
11	CEO	10	1,000,000.00

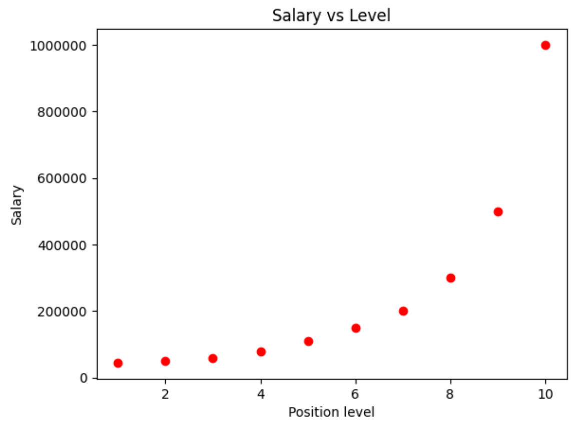
```
df= pd.read_csv('salary.csv')
```

Week3

```
x = df.iloc[:,1:2].values #level column
y = df.iloc[:,2].values #Salary column
```

Data visualization

```
plt.scatter(x,y, color='red')
plt.ticklabel_format(style='plain')
plt.title('Salary vs Level')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



predict

```
degree=['Degree1', 'Degree2', 'Degree3',
'Degree4']
Predict=pd.DataFrame(index=degree).T
Rscore = []

for k in range(1, 5):
  poly_features=PolynomialFeatures(degree=k)
  x_poly=poly_features.fit_transform(x)
  model=LinearRegression()
  model.fit(x_poly,y)
  p1=model.predict(x_poly)
  if(k==1):
    Predict.Degree1=p1
  elif(k==2):
    Predict.Degree2=p1
  elif(k==3):
    Predict.Degree3=p1
  else:
    Predict.Degree4=p1
  Rscore.append(model.score(x_poly,y))
```

Week3

Predict

	Degree1	Degree2	Degree3	Degree4
0	-114454.545455	118727.272727	14902.097902	53356.643357
1	-33575.757576	44151.515152	78759.906760	31759.906760
2	47303.030303	8439.393939	94960.372960	58642.191142
3	128181.818182	11590.909091	88223.776224	94632.867133
4	209060.606061	53606.060606	83270.396270	121724.941725
5	289939.393939	134484.848485	104820.512821	143275.058275
6	370818.181818	254227.272727	177594.405594	184003.496504
7	451696.969697	412833.333333	326312.354312	289994.172494
8	532575.757576	610303.030303	575694.638695	528694.638695
9	613454.545455	846636.363636	950461.538462	988916.083916

Rscore

[0.6690412331929895,
0.9162082221443942,
0.9812097727913366,
0.9973922891706614]

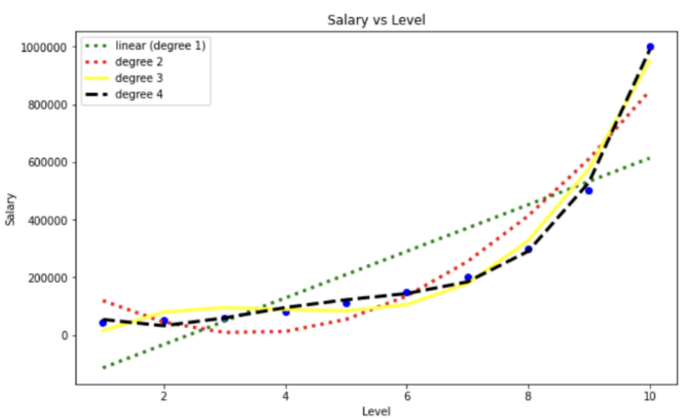
Model

```
print(model.intercept_, model.coef_)

184166.66666719783 [      0.         -211002.33100292   94765.44289063
-15463.28671331
     890.15151515]
```

Prediction

```
x_poly = PolynomialFeatures(degree=4)
model.predict(x_poly.fit_transform([[6.5]]))

array([158862.45265155])
```

Visualization of model prediction



Naïve Bayes Classifier

Naïve Bayes Classifier / Case study on Loan Prediction

Dataset: `simple_loan.csv`

age	employed	own_house	credit	target
young	FALSE	n	fair	no
young	FALSE	n	good	no
young	TRUE	n	good	yes
young	TRUE	y	fair	yes
young	FALSE	n	fair	no
middle	FALSE	n	fair	no
middle	FALSE	n	good	no
middle	TRUE	y	good	yes
middle	FALSE	y	excellent	yes
middle	FALSE	y	excellent	yes
old	FALSE	y	excellent	yes
old	FALSE	y	good	yes
old	TRUE	n	good	yes
old	TRUE	n	excellent	yes
old	FALSE	n	fair	no
old	FALSE	n	excellent	yes
young	TRUE	y	fair	yes

Naïve Bayes Classifier (Manual Computing)

age	employed	own_house	credit	target
young	FALSE	n	fair	no
young	FALSE	n	good	no
young	TRUE	n	good	yes
young	TRUE	y	fair	yes
young	FALSE	n	fair	no
middle	FALSE	n	fair	no
middle	FALSE	n	good	no
middle	TRUE	y	good	yes
middle	FALSE	y	excellent	yes
middle	FALSE	y	excellent	yes
old	FALSE	y	excellent	yes
old	FALSE	y	good	yes
old	TRUE	n	good	yes
old	TRUE	n	excellent	yes
old	FALSE	n	fair	no
old	FALSE	n	excellent	yes
young	TRUE	y	fair	yes

P(target = "no") = 6/17 = 0.3529 **P(target = "yes") = 11/17= 0.6471**

P(age = "middle" | target = "no") = 2/6 P(age = "middle" | target = "yes")= 3/11
P(age = "old" | target = "no") = 1/6 P(age = "old" | target = "yes") = 5/11
P(age = "young" | target = "no") = 3/6 P(age = "young" | target = "yes") = 3/11

P(employed="false" | target="no") = 6/6 P(employed="false" | target="yes") = 5/11
P(employed="true" | target="no") = 0/6 P(employed="true" | target="yes") = 6/11

P(own_house = "n" | target="no") = 6/6 P(own_house = "n" | target= "yes") = 4/11
P(own_house = "y" | target="no") = 0/6 P(own_house = "y" | target="yes") = 7/11

P(credit= "excellent" | target="no") = 0/6 P(credit= "excellent" | target="yes") = 5/11
P(credit= "fair" | target="no") = 4/6 P(credit= "fair" | target="yes") = 2/11
P(credit= "good" | target="no") = 2/6 P(credit= "good" | target="yes") = 4/11

6

Prediction a New Customer

- a new customer X
- X = (age ="old", employed = "false", own_house = "n", credit= "good")

P(target = "no") = 6/17 = 0.3529 **P(target = "yes") = 11/17= 0.6471**

P(age = "old" | target = "no") = 1/6
P(age = "old" | target = "yes") = 5/11

P(employed="false" | target="no") = 6/6
P(employed="false" | target="yes") = 5/11

P(own_house = "n" | target="no") = 6/6
P(own_house = "n" | target= "yes") = 4/11

P(credit= "good" | target="no") = 2/6
P(credit= "good" | target="yes") = 4/11

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When v_j= target="no"

= (6/17) x (1/6) x (6/6) x (6/6) x (2/6) = 0.019608

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When v_j= target="yes"

= (11/17) x (5/11) x (5/11) x (4/11) x (4/11) = 0.017678

Therefore, X belongs to class ("target= no")

7

Prediction a New Customer

- a new customer X
- X = (age ="middle", employed = "true", own_house = "y", credit= "fair")

P(target = "no") = 6/17 = 0.3529 **P(target = "yes") = 11/17= 0.6471**

P(age = "middle" | target = "no") = 2/6
P(age = "middle" | target = "yes")= 3/11
P(employed="true" | target="no") = 0/6
P(employed="true" | target="yes") = 6/11
P(own_house = "y" | target="no") = 0/6
P(own_house = "y" | target="yes") = 7/11
P(credit= "fair" | target="no") = 4/6
P(credit= "fair" | target="yes") = 2/11

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When v_j= target="no"

= (6/17) x (2/6) x 0 x 0 x (4/6) = 0

$\hat{P}(v_j)\prod_{i=1}^{n} P(a_i|v_j)$ When v_j= target="yes"

= (11/17) x (3/11) x (6/11) x (7/11) x (2/11) = 0.011137

Therefore, X belongs to class ("target= yes")

Week4

Python Programming for Loan Prediction

Upload and Read Data File

```
df= pd.read_csv('simple_loan.csv')
X=df.drop(['target'], axis=1)
y=df.target
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
def labelEncode(data,columns):
  for i in columns:
    lb = LabelEncoder().fit_transform(data[i])
    data[i+'_'] = lb #คอลัมที่ encode จะเป็นชื่อ + '_'
```

```
f_columns=['age', 'employed','own_house', 'credit']
labelEncode(X,f_columns) #ใช้ function Encode

y_le = LabelEncoder()
y1 = y_le.fit_transform(y)
```

เลือกเฉพาะที่ encode แล้วมาใส่ใน X1

<u>เลขที่ encode จะเรียงตามตัวอักษร</u>

```
X1=X[['age_', 'employed_','own_house_', 'credit_']]
```

X

	age	employed	own_house	credit	age_	employed_	own_house_	credit_
0	young	False	n	fair	2	0	0	1
1	young	False	n	good	2	0	0	2
2	young	True	n	good	2	1	0	2
3	young	True	y	fair	2	1	1	1
4	young	False	n	fair	2	0	0	1
5	middle	False	n	fair	0	0	0	1
6	middle	False	n	good	0	0	0	2
7	middle	True	y	good	0	1	1	2
8	middle	False	y	excellent	0	0	1	0
9	middle	False	y	excellent	0	0	1	0
10	old	False	y	excellent	1	0	1	0
11	old	False	y	good	1	0	1	2
12	old	True	n	good	1	1	0	2
13	old	True	n	excellent	1	1	0	0
14	old	False	n	fair	1	0	0	1
15	old	False	n	excellent	1	0	0	0
16	young	True	y	fair	2	1	1	1

X1

	age_	employed_	own_house_	credit_
0	2	0	0	1
1	2	0	0	2
2	2	1	0	2
3	2	1	1	1
4	2	0	0	1
5	0	0	0	1
6	0	0	0	2
7	0	1	1	2
8	0	0	1	0
9	0	0	1	0
10	1	0	1	0
11	1	0	1	2
12	1	1	0	2
13	1	1	0	0
14	1	0	0	1
15	1	0	0	0
16	2	1	1	1

Model Construction

```
from sklearn.naive_bayes import CategoricalNB
model = CategoricalNB()
model.fit(X1,y1)
```

```
print(model.feature_log_prob_)

[array([[-1.09861229, -1.5040774 , -0.81093022],    Log(P(age=middle|target=no))
        [-1.25276297, -0.84729786, -1.25276297]]),   = -1.09861229
array([[-0.13353139, -2.07944154],
        [-0.77318989, -0.61903921]]),
array([[-0.13353139, -2.07944154],
        [-0.95551145, -0.48550782]]),
array([[-2.19722458, -0.58778666, -1.09861229],
        [-0.84729786, -1.54044504, -1.02961942]])]
```

```
print(model.category_count_)

age                                     Count(age=... && target=...) = …
[array([[2., 1., 3.],                   target = no / middle old young
        [3., 5., 3.]]),                 target = yes / middle old young
employed
array([[6., 0.],                        target = no / false
        [5., 6.]]),                     target = yes / true
own_house
array([[6., 0.],                        target = no / n
        [4., 7.]]),                     target = yes / y
credit
array([[0., 4., 2.],                    target = no / excellent fair good
        [5., 2., 4.]])]                 target = yes / excellent fair good
```

Week4

Model Prediction

1. age =“middle”, employed = “true”, own_house =“y”, credit= “fair” ⇒ (0 1 1 1)
2. age =“old”, employed = “false”, own_house =“n”, credit= “good” ⇒ (1 0 0 2)

```
new_input=[[0,1,1,1],[1,0,0,2]]
y_prob_pred = model.predict_proba(new_input)
```

ดูแบบปริ้นผลลัพธ์

```
y_new_predict=model.predict(new_input)
n=1
for i in y_new_predict:
  print( 'No' ,n, '=>: ',y_le.classes_[i])
  n=n+1

No 1 =>:  yes // คนที่1
No 2 =>:  no // คนที่2
```

ดูแบบเทียบเอง

```
y_prob_pred

array([[0.0721808 , 0.9278192 ],       คนแรก target yes > no

       [0.53238717, 0.46761283]])      คนที่2 target yes < no
```

ผลการ predict คือ

คนที่1 ⇒ target = “yes” (1)

คนที่2 ⇒ target = “no” (0)

เลขจะไม่เท่าแบบคำนวณมือเพราะสูตรใน python บวกแอลฟา

Monte Carlo Simulation

Estimating the value of Pi

Monte Carlo simulations use random sampling to obtain numerical results

The Algorithm

Set the radius, sampling size to N (#iteration of random points)

circle_points=0

For i=1 to N

random x and y as a point p=(x,y)

If point p is inside the circle increment circle_points

End for

Calculate Pi = 4*(circle_points/N)

Return Pi

Checking the Position of Point p (inside or outside the circle)

If distance(P,center) $\leq$ r = inside [distance(P,center) = $\sqrt{x^2 + y^2}$]

note: center = (0,0)

Python : Estimating the value of Pi

```
import random
import math
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

set the initial values

```
r = 1.0 # radius
N = 1001 # number of iteration
d = {"Trials":[],"Pi":[]}
```

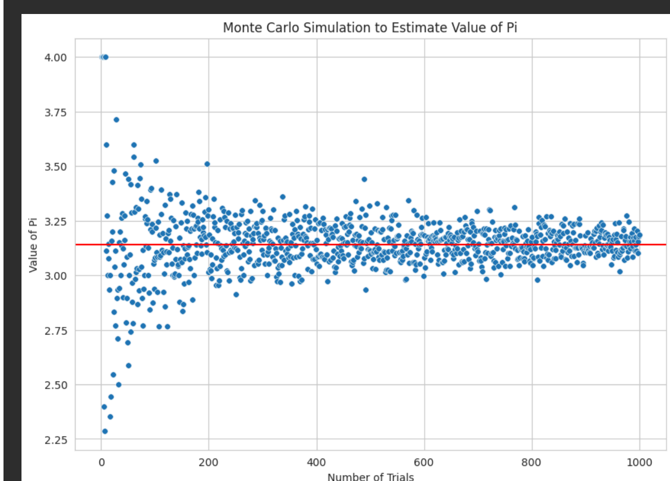
Week 5

Monte Carlo Simulation

```
for T in range(1,N):
  circle_p=0
  for i in range(T):
    x = random.uniform(-1.0, 1.0)
    y = random.uniform(-1.0, 1.0)
    x2 = x ** 2
    y2 = y ** 2

    if math.sqrt(x2 + y2) <= r:
      circle_p+=1

  d["Trials"].append(T)
  d["Pi"].append((circle_p/T)*4)
```

Visualize Pi values calculated by Monte Carlo

```
df = pd.DataFrame(data=d)
plt.figure(figsize = (10,7))
plot = sns.scatterplot(x="Trials", y="Pi", s=30, marker="o", data=df)
plot.set(title='Monte Carlo Simulation to Estimate Value of Pi', xlabel="
Number of Trials", ylabel="Value of Pi")
plt.axhline(y=3.14, color='r', linestyle='-')
plt.show()
```

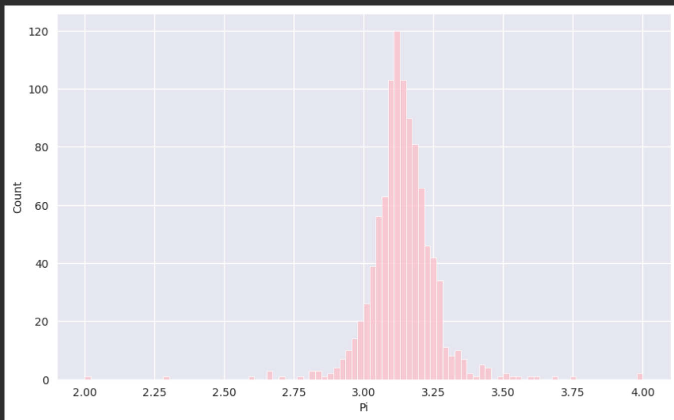


Histogram of Pi values

```
sns.set_style('darkgrid')
fig = plt.figure(figsize = (10,6))
```

Week 5

```
sns.histplot(df, x="Pi",color='pink');
```

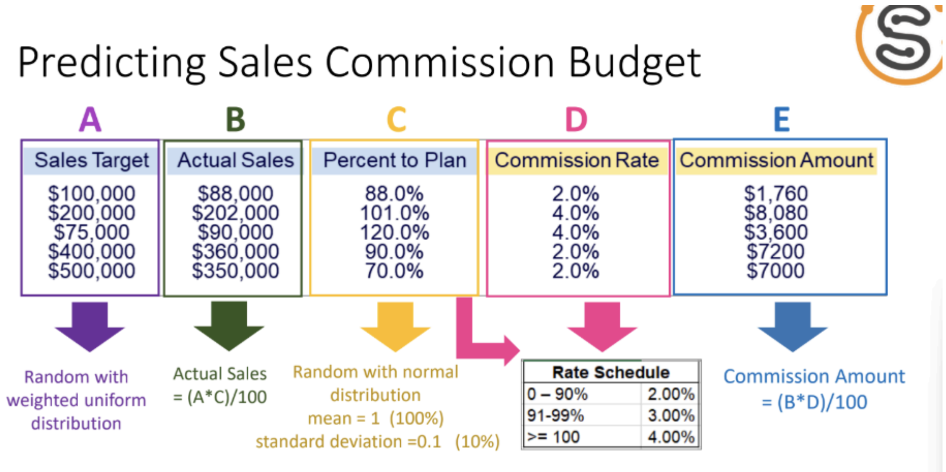



```
df['Pi'].mean()

3.138038884790763
```

Mean ของ pi แต่ละรอบจะได้ไม่เท่ากัน เพราะเป็นการ random

Week 5

Predicting sales commission budget



Random **C** → Random **A** → Cal **B**,**D** → Cal **E**

Python : Predicting sales commission budget

```
import pandas as pd
import numpy as np
import seaborn as sns
sns.set_style('whitegrid')
```

```
avg = 1
std_dev = .1
num_reps = 500 #จน.พนง
```

```
sales_target_values = [75_000, 100_000, 200_000, 300_000, 400_000, 500_000]
sales_target_prob = [.3, .3, .2, .1, .05, .05]  #+กันแล้วได้1
```

Calculate Commission Rate function

```
def calc_commission_rate(x):
  if x <= .90:
    return .02
  if x <= .99:
    return .03
  else:
    return .04
```

Week 5

Simulation with 1,000 iterations

```
num_simulations = 1000
all_stats = []
# Loop through many simulations
for i in range(num_simulations):

  sales_target = np.random.choice(sales_target_values, num_reps,
                p=sales_target_prob)
  #Random Sales Target (weighted uniform distribution)
  pct_to_target = np.random.normal(avg, std_dev, num_reps).round(2)
  #Random Percent to Plan (normal distribution)

  # สร้าง dataframe based on the inputs and number of reps
  df = pd.DataFrame(index=range(num_reps) , data={'Pct_To_Target':
    pct_to_target,'Sales_Target': sales_target})

  # คำนวน Actual Sale
  df['Sales'] = df['Pct_To_Target'] * df['Sales_Target']

  # Determine the commissions rate and calculate it
  df['Commission_Rate'] = df['Pct_To_Target'].apply(calc_commission_rate)
  df['Commission_Amount'] = df['Commission_Rate'] * df['Sales']

  # We want to track sales,commission amounts and sales targets over all the
simulations
  all_stats.append([df['Sales'].sum().round(0),
                  df['Commission_Amount'].sum().round(0),
                  df['Sales_Target'].sum().round(0)])
  # sumรอบละ500คน ทั้งหมด1000รอบ
```

```
results_df = pd.DataFrame.from_records(all_stats,
columns=['Sales','Commission_Amount','Sales_Target'])
results_df
```

	Sales	Commission_Amount	Sales_Target
0	82492750.0	2814232.0	82975000
1	84638250.0	2932080.0	84050000
2	84118000.0	2882388.0	84025000
3	83171750.0	2831272.0	83475000
4	84356000.0	2833640.0	84650000
...	...	...	...
995	80065000.0	2742478.0	80050000
996	81377250.0	2742122.0	81800000
997	84699500.0	2885540.0	84525000
998	81348250.0	2783185.0	81775000
999	82562000.0	2815660.0	82700000

1000 rows × 3 columns

Week 5

```
results_df.describe().style.format('{:,.2f}')
```

	Sales	Commission_Amount	Sales_Target
count	1,000.00	1,000.00	1,000.00
mean	83,853,300.25	2,863,612.69	83,843,300.00
std	2,765,438.83	104,464.41	2,718,974.15
min	74,864,250.00	2,541,080.00	75,025,000.00
25%	81,888,625.00	2,788,941.25	82,043,750.00
50%	83,901,125.00	2,861,297.50	83,875,000.00
75%	85,816,312.50	2,936,809.25	85,725,000.00
max	92,766,000.00	3,182,838.00	92,875,000.00

```
results_df['Commission_Amount'].plot(kind='hist', title="Total Commission Amount")
```