```
#Working with Simple Linear Regres
#Risk analysis of space shuttle
```

```
import os
os.chdir("C:/Users/Mohan/Desktop/D
```

```
---------------------------
---------------------------
---------------------
FileNotFoundError
Traceback (most recent call
last)
<ipython-input-1-
bd661e9843f5> in <module>
      1 import os
----> 2
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
challenger=pd.read_csv('challenger
```

```
challenger
```

Executions generic.py ×

```
        After regular attribute access, try lookin
        This allows simpler access to columns for
        """
        # Note: obj.x will always call obj.__getat
        # calling obj.__getattr__('x').
        if (
            name not in self._internal_names_set
            and name not in self._metadata
            and name not in self._accessors
            and self._info_axis._can_hold_identifi
        ):
            return self[name]
········return·object.__getattribute__(self,·name)

    def __setattr__(self, name: str, value) -> Non
        """
        After regular attribute access, try settin
        This allows simpler access to columns for
        """
        # first try regular attribute access via _
        # e.g. ``obj.x`` and ``obj.x = 4`` will al
        # the same attribute.

        try:
            object.__getattribute__(self, name)
                              tattr__(self, name,
```

	o_ring_ct	O.ring.failu
0	6	
1	6	
2	6	
3	6	
4	6	
5	6	
6	6	
7	6	

```
#Assign ring failure values to an
O_ring_failures=challenger['O.ring
```

10 6

```
O_ring_failures
```

```
0     0
1     1
2     0
3     0
4     0
5     0
6     0
7     0
8     1
9     1
10    1
11    0
12    0
13    2
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    1
Name: O.ring.failures,
dtype: int64
```

```
#Assign temperature values to 'tem
```

```
temp=challenger['temperature']
```

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(temp,O_ring_failures,'o')
plt.ylabel("O_ring_failures")
plt.xlabel("Temperature")
```

```
temp
```

```
0     66
1     70
2     69
3     68
4     67
5     72
6     73
7     70
8     57
9     63
10    70
11    78
12    67
13    53
14    67
15    75
16    70
17    81
18    76
19    79
20    75
21    76
22    58
Name: temperature, dtype:
int64
```

```
challenger.corr()
```

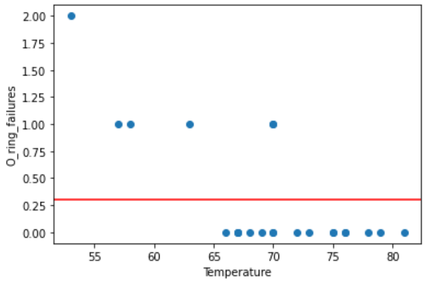
	o_ring_ct	O.
o_ring_ct	NaN	
O.ring.failures	NaN	
temperature	NaN	
pressure	NaN	
launch_id	NaN	

```
mean_O_ring_failures = challenger[
```

```
mean_O_ring_failures
```

```
0.30434782608695654
```

```
plt.plot(temp,O_ring_failures,'o')
plt.ylabel("O_ring_failures")
plt.xlabel("Temperature")
plt.axhline(mean_O_ring_failures,
plt.show()
```



```
import statsmodels.api as sm
model=sm.OLS(O_ring_failures,temp)
```

```
#Obtain model summary
model.summary()
```

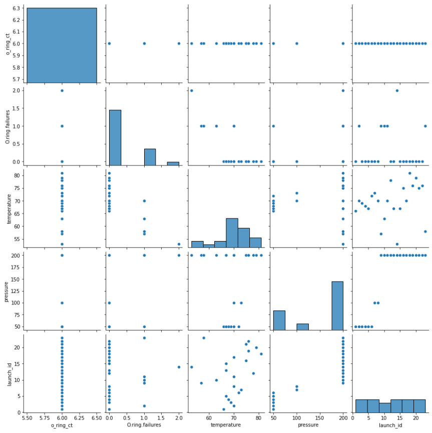
```
                      OLS Regression Re
Dep. Variable:  O.ring.failures    R
                                   (un
Model:          OLS                s
                                   (un
Method:         Least              F-
                Squares
Date:           Mon, 10 Oct        I
                2022               s
Time:           05:51:47           Li
No.             23
Observations:
Df Residuals:   22
Df Model:       1
Covariance      nonrobust
Type:
                     std
            coef           t
                     err
temperature 0.0038 0.002 2.181
                            Durbin
```

Observation

As temeparture increases by 1 degree, O.ring.failures increase by 0.0038

```
import seaborn as sns
sns.pairplot(challenger)
```

```
<seaborn.axisgrid.PairGrid
at 0x7f5fa29a0dd0>
```

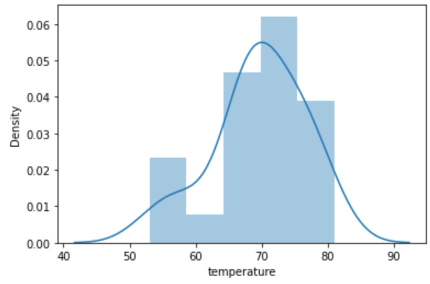


Observations

1.The histogram on the diagonal allows us to see the distribution of a single variable 2.The scatter plots on the upper and lower triangles show the relationship (or lack thereof) between two variables.

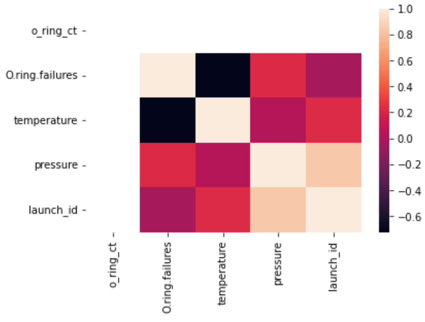
```
sns.distplot(challenger['temperatu
```

```
    /usr/local/lib/python3.7/dis
      warnings.warn(msg, FutureW
    <matplotlib.axes._subplots.A
     at 0x7f5fa1000050>
```



```
# Plot the correlation using heatm
corr = challenger.corr()
sns.heatmap(corr,xticklabels=corr.
```

```
    <matplotlib.axes._subplots.A
     at 0x7f5fa0f8ab10>
```



Observations

Black color represents negative correlation which exists betewen temeprature and O.ring.failures

[ ] ↳ 12 cells hidden

Linear Regression with sklearn

[ ] ↳ *3 cells hidden*

Observation

The slope value -0.051 means that the predicted O.ring.failures reduces by -0.05 when x(temperature) rises by one degree

```
#Check model score
model

     LinearRegression()
```

```
X_test

     array([[75],
            [81],
            [68],
            [53],
            [79],
            [70],
            [70],
            [67],
            [69],
            [76]])
```

```
X_test.shape

     (10, 1)
```

```
y_test.shape

     (10,)
```

```
X_test = X_test.values.reshape((-1
#Predictions from the model
predictions = model.predict(X_test
print('predicted O.ring.failures:'

y_test
```

```
#Visualize the predictions
plt.scatter(y_test, predictions)
```

Observations

A linear model has been obtained

```
#Other way for prediction
y_pred = model.intercept_ + model.
print('predicted response:', y_pre
```

```
predicted response:
[[-0.02417649]
 [-0.33605319]
 [ 0.33967966]
 [ 1.11937141]
 [-0.23209429]
 [ 0.23572076]
 [ 0.23572076]
 [ 0.39165911]
 [ 0.28770021]
 [-0.07615594]]
```

```
#Define new data instance
Xnew = [[30]]

#Make a Prediction
ynew = model.predict(Xnew)

#Show the inputs and predicted out
print("New Temperature=%s, Predict
```

```
New Temperature=[[30]], Pred
```

```
#Define new data instance
Xnew = [[70]]

#Make a Prediction
ynew = model.predict(Xnew)

#Show the inputs and predicted out
print("New Temperature=%s, Predict
New Temperature=[[70]], Pred
```

```
#Evaluating the model
from sklearn.metrics import mean_s
X_train = X_train.reshape(-1,1)
y_train_prediction = model.predict

X_test = X_test.reshape(-1,1)
y_test_prediction = model.predict(


# printing values
print('Slope:' ,model.coef_)
print('Intercept:', model.intercep
print("\n")

# model evaluation for training se
import numpy as np
rmse_training = (np.sqrt(mean_squa
r2_training = r2_score(y_train, y_

print("The model performance for t
print("--------------------------
print('RMSE is {}'.format(rmse_tra
print('R2 score is {}'.format(r2_t
print("\n")

# model evaluation for testing set
rmse_testing = (np.sqrt(mean_squar
r2_testing = r2_score(y_test, y_te

print("The model performance for t
print("--------------------------
print('Root mean squared error: ',
print('R2 score: ', r2_testing)
```

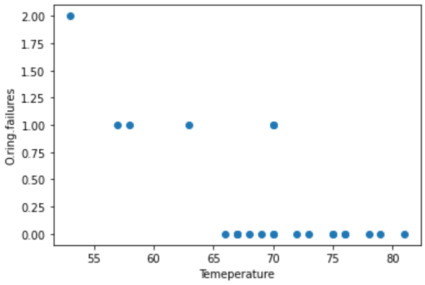
```
Slope: [-0.05197945]
Intercept: 3.874282260501661


The model performance for tr
----------------------------
RMSE is 0.32743461522828027
R2 score is 0.49669252207783


The model performance for te
----------------------------
Root mean squared error:  0.
R2 score:  0.526846824309108
```

```
# plotting values
# data points
plt.scatter(X, y)
plt.xlabel('Temeperature')
plt.ylabel('O.ring.failures')
```

```
Text(0, 0.5,
'O.ring.failures')
```



```
X= X.reshape(-1,1)
y_predicted = model.predict(X)
```

```
---------------------------
---------------------------
---------------------
AttributeError
Traceback (most recent call
last)
<ipython-input-81-
4f01f4a3466f> in <module>
----> 1 X= X.reshape(-1,1)
      2 y_predicted =
model.predict(X)

/usr/local/lib/python3.7/dis
packages/pandas/core/generic
 in __getattr__(self, name)
   5485            ):
   5486                 return
```

```
# predicted values
plt.plot(X, y_predicted, color='r'
plt.show()
```

Multiple Linear Regression

[ ] ↳ *2 cells hidden*

Observation

1.This model has a higher R-squared compared to simple linear model against temperature and O.ring.failure 2.However in this model both temeprature and pressure features became statistcially insignificant to predict O.ring.failure 3.As pressure increases by 1 atmosphere, O.ring.failures increase by 0.0031 and as temperature increases by 1 degree, O.ring.failures decrease by -0.0030

[ ] ↳ *12 cells hidden*

Observation

With both temeprature and pressure the model is not linear

```
#Evaluating the model
from sklearn.metrics import mean_s

# printing values
print('Slope:' ,model.coef_)
print('Intercept:', model.intercep
print("\n")


import numpy as np
rmse = (np.sqrt(mean_squared_error
r2 = r2_score(y,predictions)

print("The model performance")
print("---------------------------
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")
```

```
Slope: [-0.05197945]
Intercept: 3.874282260501661


---------------------------
---------------------------
---------------------
ValueError
Traceback (most recent call
last)
<ipython-input-74-
e28bdfb07778> in <module>
      9
     10 import numpy as np
---> 11 rmse =
(np.sqrt(mean_squared_error(

     12 r2 =
r2_score(y,predictions)
     13

  ———— ⇕ 2 frames ————
/usr/local/lib/python3.7/dis
packages/sklearn/utils/valid
```

Colab paid products - Cancel contracts here

0s completed at 7:04 PM