Class_16.04.2023_Linear_Regression_Implementation

April 17, 2023

Linear Regression

- Predictive modelling
- Supervised Learning method
- Labels are available
- Dependent Variable , Independent Variable
- Features/attributes - Independent variable
- Target - Dependent Variable
- X is my independent variable , y is my dependent variable - their relationship is linear ir
- x is increasing , y should also increase / decrease
- x is decreasing , y should also decrease / increase
- No multi colinearity - 3 independent variables , 1 dependent variable
- a , b , c - if a is increasing it is not mandatory that b or c will have an impact on the s
- Errors - difference between actual data point and predicted data point - Residuals
- Residuals if plotted on a distribution curve , it will display a normal distribution

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
df = pd.DataFrame([['akash',23,'intern',20000],
                   ['alok',25,'intern',21000],
                   ['rahul',28,'executive',25000],
                   ['kaushik',33,'manager',30000],
                   ['shovan',29,'executive',26000],
                   ['diksha',34,'manager',32000],
                   ['aritra',23,'executive',24000],
                   ['amrita',31,'executive',26000]],columns =␣
 ↪['Name','Age','Designation','Salary'])
```

[3]:
```
df.head()
```

[3]:

	Name	Age	Designation	Salary
0	akash	23	intern	20000
1	alok	25	intern	21000
2	rahul	28	executive	25000
3	kaushik	33	manager	30000

```
4   shovan   29   executive   26000
```

[4]:
```
x = df[['Name','Age','Designation']]
y = df[['Salary']]
```

[5]:
```
x
```

[5]:
```
      Name  Age Designation
0    akash   23      intern
1     alok   25      intern
2    rahul   28   executive
3  kaushik   33     manager
4   shovan   29   executive
5   diksha   34     manager
6   aritra   23   executive
7   amrita   31   executive
```

[6]:
```
y
```

[6]:
```
   Salary
0   20000
1   21000
2   25000
3   30000
4   26000
5   32000
6   24000
7   26000
```

[7]:
```
print(len(x))
```

```
8
```

[8]:
```
x.drop(['Name'],axis=1,inplace=True)
```

```
C:\Users\sengu\AppData\Local\Temp\ipykernel_19044\181599321.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  x.drop(['Name'],axis=1,inplace=True)
```

[9]:
```
x
```

[9]:
```
   Age Designation
0   23      intern
1   25      intern
2   28   executive
```

```
3   33     manager
4   29   executive
5   34     manager
6   23   executive
7   31   executive
```

[11]:
```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
```

[12]:
```
enc_data = pd.DataFrame(enc.fit_transform(x[['Designation']]).toarray())
```

[13]:
```
x = x.join(enc_data)
```

[14]:
```
x.head()
```

[14]:

	Age	Designation	0	1	2
0	23	intern	0.0	1.0	0.0
1	25	intern	0.0	1.0	0.0
2	28	executive	1.0	0.0	0.0
3	33	manager	0.0	0.0	1.0
4	29	executive	1.0	0.0	0.0

[15]:
```
x.drop(['Designation'],axis=1 , inplace = True)
```

[16]:
```
x.head()
```

[16]:

	Age	0	1	2
0	23	0.0	1.0	0.0
1	25	0.0	1.0	0.0
2	28	1.0	0.0	0.0
3	33	0.0	0.0	1.0
4	29	1.0	0.0	0.0

[17]:
```
from sklearn.preprocessing import StandardScaler
st = StandardScaler()
```

[18]:
```
x['Age'] = np.array(x['Age']).reshape(-1,1)
```

[19]:
```
x
```

[19]:

	Age	0	1	2
0	23	0.0	1.0	0.0
1	25	0.0	1.0	0.0
2	28	1.0	0.0	0.0
3	33	0.0	0.0	1.0
4	29	1.0	0.0	0.0
5	34	0.0	0.0	1.0
6	23	1.0	0.0	0.0

```
7   31  1.0  0.0  0.0
```

[22]:
```
x.ndim
```

[22]:
```
2
```

[25]:
```
from sklearn.linear_model import LinearRegression
```

[26]:
```
from sklearn.model_selection import train_test_split
```

[27]:
```
xtrain , xtest , ytrain , ytest = train_test_split(x,y,test_size= 0.2)
```

[28]:
```
lr = LinearRegression()
model = lr.fit(xtrain,ytrain)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\DS WorkFlow\Data Is Good Class Files\Batches\2023-Classes\20th Aug␣
 ↪Batch\Class_16.04.2023_Linear_Regression_Implementation.ipynb Cell 25 in <cel␣
 ↪line: 2>()
      <a href='vscode-notebook-cell:/d%3A/DS%20WorkFlow/
 ↪Data%20Is%20Good%20Class%20Files/Batches/2023-Classes/20th%20Aug%20Batch/
 ↪Class_16.04.2023_Linear_Regression_Implementation.ipynb#X35sZmlsZQ%3D%3D?
 ↪line=0'>1</a> lr = LinearRegression()
----> <a href='vscode-notebook-cell:/d%3A/DS%20WorkFlow/
 ↪Data%20Is%20Good%20Class%20Files/Batches/2023-Classes/20th%20Aug%20Batch/
 ↪Class_16.04.2023_Linear_Regression_Implementation.ipynb#X35sZmlsZQ%3D%3D?
 ↪line=1'>2</a> model = lr.fit(xtrain,ytrain)

File d:\Anaconda\envs\github1\lib\site-packages\sklearn\linear_model\_base.py:
 ↪649, in LinearRegression.fit(self, X, y, sample_weight)
    645 n_jobs_ = self.n_jobs
    647 accept_sparse = False if self.positive else ["csr", "csc", "coo"]
--> 649 X, y = self._validate_data(
    650     X, y, accept_sparse=accept_sparse, y_numeric=True, multi_output=Tru
    651 )
    653 sample_weight = _check_sample_weight(
    654     sample_weight, X, dtype=X.dtype, only_non_negative=True
    655 )
    657 X, y, X_offset, y_offset, X_scale = _preprocess_data(
    658     X,
    659     y,
   (…)
    662     sample_weight=sample_weight,
    663 )

File d:\Anaconda\envs\github1\lib\site-packages\sklearn\base.py:518, in␣
 ↪BaseEstimator._validate_data(self, X, y, reset, validate_separately,␣
 ↪**check_params)
```

```
    453 def _validate_data(
    454     self,
    455     X="no_validation",
   (…)
    459     **check_params,
    460 ):
    461     """Validate input data and set or check the `n_features_in_`␣
 ↪attribute.
    462
    463     Parameters
   (…)
    516         validated.
    517     """
--> 518     self._check_feature_names(X, reset=reset)
    520     if y is None and self._get_tags()["requires_y"]:
    521         raise ValueError(
    522             f"This {self.__class__.__name__} estimator "
    523             "requires y to be passed, but the target y is None."
    524         )

File d:\Anaconda\envs\github1\lib\site-packages\sklearn\base.py:385, in␣
 ↪BaseEstimator._check_feature_names(self, X, reset)
    365 """Set or check the `feature_names_in_` attribute.
    366
    367 .. versionadded:: 1.0
   (…)
    381         should set `reset=False`.
    382 """
    384 if reset:
--> 385     feature_names_in = _get_feature_names(X)
    386     if feature_names_in is not None:
    387         self.feature_names_in_ = feature_names_in

File d:\Anaconda\envs\github1\lib\site-packages\sklearn\utils\validation.py:
 ↪1893, in _get_feature_names(X)
   1891 # mixed type of string and non-string is not supported
   1892 if len(types) > 1 and "str" in types:
-> 1893     raise TypeError(
   1894         "Feature names are only supported if all input features have␣
 ↪string names, "
   1895         f"but your input has {types} as feature name / column name type
 ↪ "
   1896         "If you want feature names to be stored and validated, you must
 ↪convert "
   1897         "them all to strings, by using X.columns = X.columns.astype(str␣
 ↪for "
   1898         "example. Otherwise you can remove feature / column names from␣
 ↪your input "
```

```
   1899         "data, or convert them all to a non-string data type."
   1900     )
   1902 # Only feature names of all strings are supported
   1903 if len(types) == 1 and types[0] == "str":

TypeError: Feature names are only supported if all input features have string␣
 ↪names, but your input has ['int', 'str'] as feature name / column name types.␣
 ↪If you want feature names to be stored and validated, you must convert them␣
 ↪all to strings, by using X.columns = X.columns.astype(str) for example.␣
 ↪Otherwise you can remove feature / column names from your input data, or␣
 ↪convert them all to a non-string data type.
```

[ ]: