students_mark_predictor

November 18, 2022

[29]:
```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1 Load Dataset

[2]:
```
path = r"https://drive.google.com/uc?
 ↪export=download&id=13ZTYmL3E8S0nz-UKl4aaTZJaI3DVBGHM"
df   = pd.read_csv(path)
```

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

[3]:

	study_hours	student_marks
0	6.83	78.50
1	6.56	76.74
2	NaN	78.68
3	5.67	71.82
4	8.67	84.19

[4]:
```
df.tail()
```

[4]:

	study_hours	student_marks
195	7.53	81.67
196	8.56	84.68
197	8.94	86.75
198	6.60	78.05
199	8.35	83.50

[5]:
```
df.shape
```

[5]: (200, 2)

1.1 Discover and visualize the data to gain insights

[6]:
```
df.info()
```

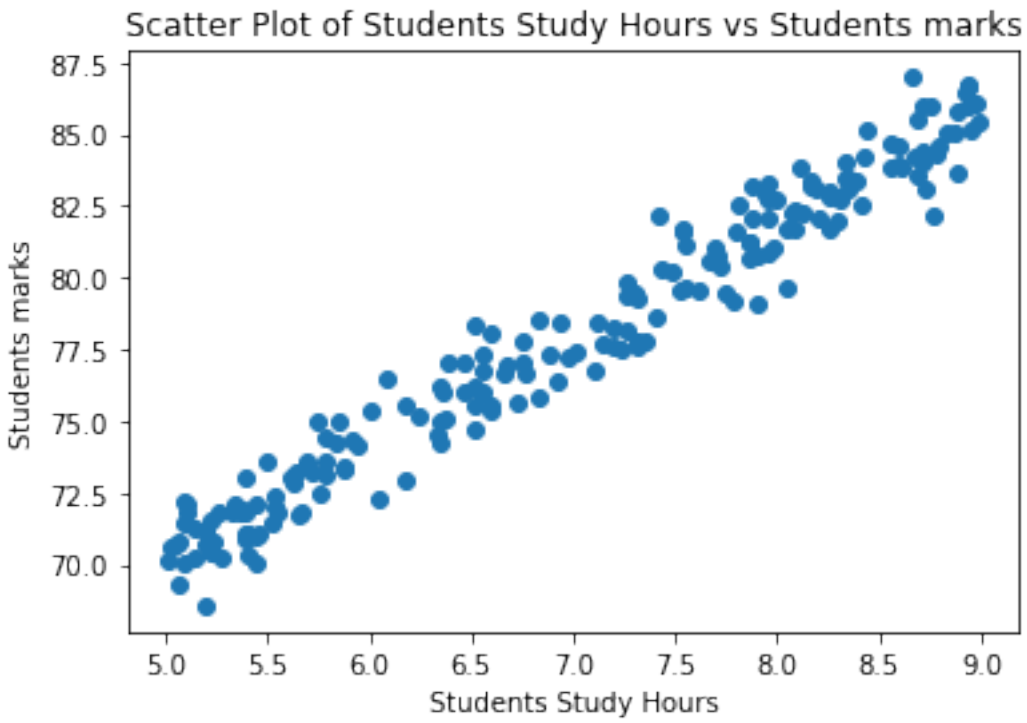
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   study_hours    195 non-null    float64
 1   student_marks  200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

[7]:
```
df.describe()
```

[7]:

	study_hours	student_marks
count	195.000000	200.00000
mean	6.995949	77.93375
std	1.253060	4.92570
min	5.010000	68.57000
25%	5.775000	73.38500
50%	7.120000	77.71000
75%	8.085000	82.32000
max	8.990000	86.99000

[8]:
```
plt.scatter(x =df.study_hours, y = df.student_marks)
plt.xlabel("Students Study Hours")
plt.ylabel("Students marks")
plt.title("Scatter Plot of Students Study Hours vs Students marks")
plt.show()
```



1.2 Prepare the data for Machine Learning algorithms

[9]:
```
# Data Cleaning
```

[10]:
```
df.isnull().sum()
```

[10]:
```
study_hours      5
student_marks    0
dtype: int64
```

[11]:
```
df.mean()
```

[11]:
```
study_hours       6.995949
student_marks    77.933750
dtype: float64
```

[12]:
```
df2 = df.fillna(df.mean())
```

[13]:
```
df2.isnull().sum()
```

[13]:
```
study_hours      0
student_marks    0
dtype: int64
```

[14]:
```
df2.head()
```

[14]:

	study_hours	student_marks
0	6.830000	78.50
1	6.560000	76.74
2	6.995949	78.68
3	5.670000	71.82
4	8.670000	84.19

[15]:
```
# split dataset
```

[16]:
```
X = df2.drop("student_marks", axis = "columns")
y = df2.drop("study_hours", axis = "columns")
print("shape of X = ", X.shape)
print("shape of y = ", y.shape)
```

```
shape of X =  (200, 1)
shape of y =  (200, 1)
```

[17]:
```
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test = train_test_split(X,y, test_size = 0.2,␣
 ↪random_state=51)
print("shape of X_train = ", X_train.shape)
print("shape of y_train = ", y_train.shape)
print("shape of X_test = ", X_test.shape)
print("shape of y_test = ", y_test.shape)
```

```
shape of X_train =  (160, 1)
shape of y_train =  (160, 1)
shape of X_test =  (40, 1)
shape of y_test =  (40, 1)
```

2 Select a model and train it

[18]:
```
# y = m * x + c
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

[19]:
```
lr.fit(X_train,y_train)
```

[19]:
```
LinearRegression()
```

[20]:
```
lr.coef_
```

[20]:
```
array([[3.93571802]])
```

[21]:
```
lr.intercept_
```

[21]:
```
array([50.44735504])
```

[22]:
```
m = 3.93
c = 50.44
y  = m * 4 + c
y
```

[22]:
```
66.16
```

[23]:
```
lr.predict([[4]])[0][0].round(2)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

[23]:
```
66.19
```

[24]:
```
y_pred  = lr.predict(X_test)
y_pred
```

[24]:
```
array([[83.11381458],
       [78.9025963 ],
       [84.57003024],
       [85.82946001],
       [84.72745896],
       [80.75238377],
       [72.84159055],
       [71.66087515],
       [73.23516235],
       [71.66087515],
       [73.47130543],
       [76.38373677],
       [73.23516235],
       [73.58937697],
       [82.95638585],
       [70.40144538],
       [73.23516235],
       [78.74516758],
       [75.55723598],
       [82.68088559],
       [76.65923703],
       [70.48015974],
       [74.77009238],
       [77.98143645],
       [85.59331693],
       [82.56281405],
       [76.42309395],
```

```
       [85.0423164 ],
       [78.39095296],
       [81.38209865],
       [81.73631327],
       [83.15317176],
       [82.20859943],
       [81.10659839],
       [73.58937697],
       [71.1492318 ],
       [71.89701823],
       [81.53952737],
       [72.60544747],
       [71.93637541]])
```

[25]:
```
pd.DataFrame(np.c_[X_test, y_test, y_pred], columns = ["study_hours",␣
↪"student_marks_original","student_marks_predicted"])
```

[25]:

	study_hours	student_marks_original	student_marks_predicted
0	8.300000	82.02	83.113815
1	7.230000	77.55	78.902596
2	8.670000	84.19	84.570030
3	8.990000	85.46	85.829460
4	8.710000	84.03	84.727459
5	7.700000	80.81	80.752384
6	5.690000	73.61	72.841591
7	5.390000	70.90	71.660875
8	5.790000	73.14	73.235162
9	5.390000	73.02	71.660875
10	5.850000	75.02	73.471305
11	6.590000	75.37	76.383737
12	5.790000	74.44	73.235162
13	5.880000	73.40	73.589377
14	8.260000	81.70	82.956386
15	5.070000	69.27	70.401445
16	5.790000	73.64	73.235162
17	7.190000	77.63	78.745168
18	6.380000	77.01	75.557236
19	8.190000	83.08	82.680886
20	6.660000	76.63	76.659237
21	5.090000	72.22	70.480160
22	6.180000	72.96	74.770092
23	6.995949	76.14	77.981436
24	8.930000	85.96	85.593317
25	8.160000	83.36	82.562814
26	6.600000	78.05	76.423094
27	8.790000	84.60	85.042316
28	7.100000	76.76	78.390953

29	7.860000	81.24	81.382099
30	7.950000	80.86	81.736313
31	8.310000	82.69	83.153172
32	8.070000	82.30	82.208599
33	7.790000	79.17	81.106598
34	5.880000	73.34	73.589377
35	5.260000	71.86	71.149232
36	5.450000	70.06	71.897018
37	7.900000	80.76	81.539527
38	5.630000	72.87	72.605447
39	5.460000	71.10	71.936375

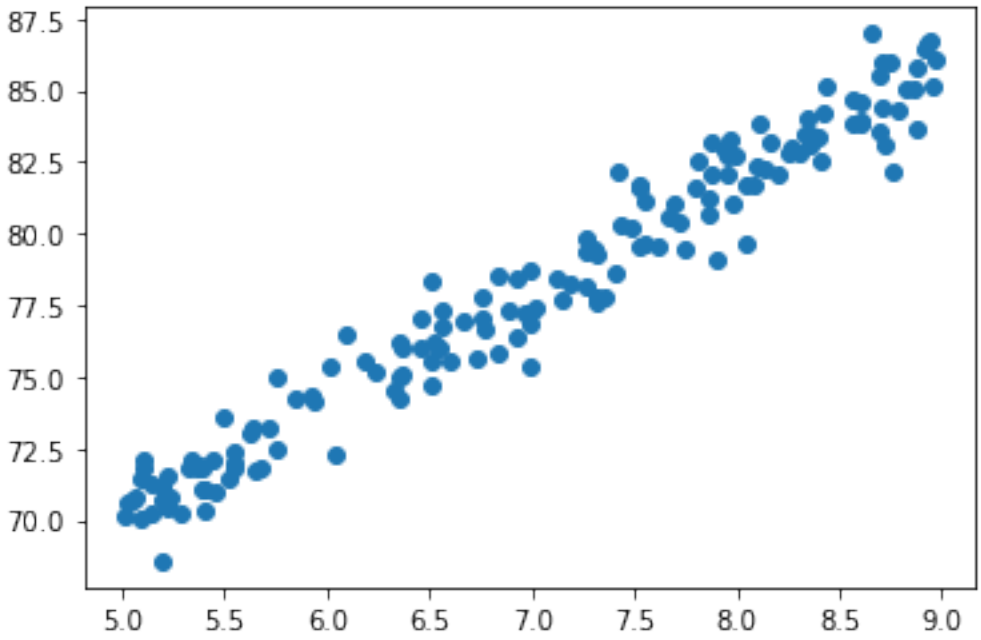
2.1 Fine-tune the model

[26]:
```
lr.score(X_test,y_test)
```

[26]:
```
0.9514124242154464
```

[27]:
```
plt.scatter(X_train,y_train)
```

[27]:
```
<matplotlib.collections.PathCollection at 0x2c4a2c3d1c0>
```



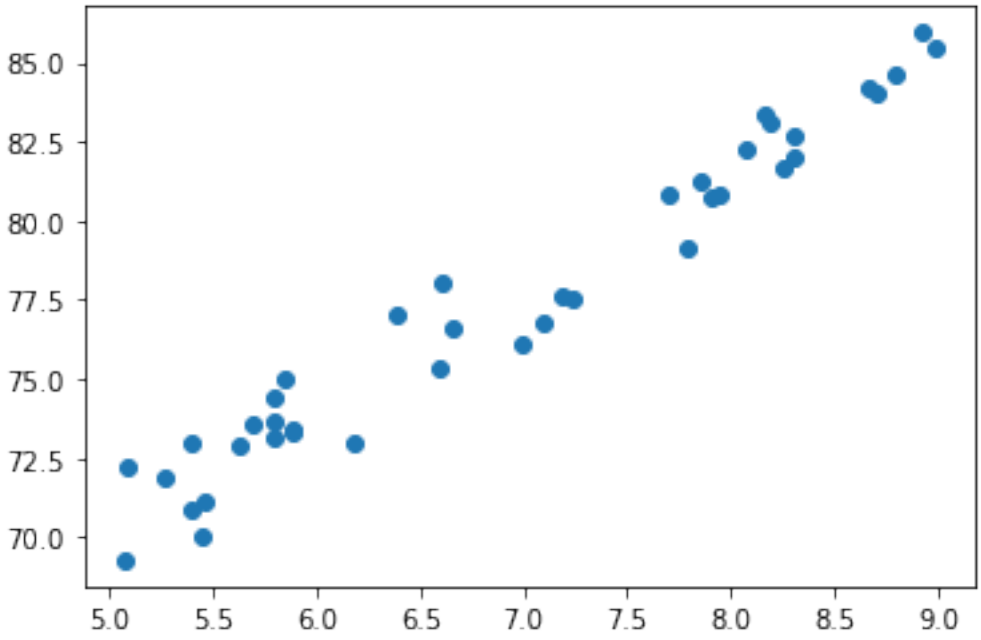
[28]:
```
plt.scatter(X_test, y_test)
plt.plot(X_train, lr.predict(X_train), color = "r")
```

```
----------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)
File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:
 ↪3621, in Index.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in␣
 ↪pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx:142, in␣
 ↪pandas._libs.index.IndexEngine.get_loc()

TypeError: '(slice(None, None, None), None)' is an invalid key

During handling of the above exception, another exception occurred:

InvalidIndexError                         Traceback (most recent call last)
Input In [28], in <cell line: 2>()
      1 plt.scatter(X_test, y_test)
----> 2 plt.plot(X_train, lr.predict(X_train), color = "r")

File C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\pyplot.py:2757, in␣
 ↪plot(scalex, scaley, data, *args, **kwargs)
   2755 @_copy_docstring_and_deprecators(Axes.plot)
   2756 def plot(*args, scalex=True, scaley=True, data=None, **kwargs):
-> 2757     return gca().plot(
   2758         *args, scalex=scalex, scaley=scaley,
   2759         **({"data": data} if data is not None else {}), **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:1632,␣
 ↪in Axes.plot(self, scalex, scaley, data, *args, **kwargs)
   1390 """
   1391 Plot y versus x as lines and/or markers.
   1392
   (…)
   1629 (``'green'``) or hex strings (``'#008000'``).
   1630 """
   1631 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1632 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633 for line in lines:
   1634     self.add_line(line)

File C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:312, i␣
 ↪_process_plot_var_args.__call__(self, data, *args, **kwargs)
    310     this += args[0],
    311     args = args[1:]
--> 312 yield from self._plot_args(this, kwargs)
```

```
File C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:487, i
 ↪_process_plot_var_args._plot_args(self, tup, kwargs, return_kwargs)
    484         kw[prop_name] = val
    486 if len(xy) == 2:
--> 487     x = _check_1d(xy[0])
    488     y = _check_1d(xy[1])
    489 else:

File C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py:
 ↪1327, in _check_1d(x)
   1321 with warnings.catch_warnings(record=True) as w:
   1322     warnings.filterwarnings(
   1323         "always",
   1324         category=Warning,
   1325         message='Support for multi-dimensional indexing')
-> 1327     ndim = x[:, None].ndim
   1328     # we have definitely hit a pandas index or series object
   1329     # cast to a numpy array.
   1330     if len(w) > 0:

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:3505, in
 ↪DataFrame.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:
 ↪3628, in Index.get_loc(self, key, method, tolerance)
   3623         raise KeyError(key) from err
   3624     except TypeError:
   3625         # If we have a listlike key, _check_indexing_error will raise
   3626         #  InvalidIndexError. Otherwise we fall through and re-raise
   3627         #  the TypeError.
-> 3628         self._check_indexing_error(key)
   3629         raise
   3631 # GH#42269

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py:
 ↪5637, in Index._check_indexing_error(self, key)
   5633 def _check_indexing_error(self, key):
   5634     if not is_scalar(key):
   5635         # if key is not a scalar, directly raise an error (the code bel w
   5636         # would convert to numpy arrays and raise later any way) -
 ↪GH29926
-> 5637         raise InvalidIndexError(key)
```

```
InvalidIndexError: (slice(None, None, None), None)
```



2.2 Save Ml Model

[ ]:
```
import joblib
joblib.dump(lr, "student_mark_predictor.pkl")
```

[ ]:
```
model = joblib.load("student_mark_predictor.pkl")
```

[ ]:
```
model.predict([[5]])[0][0]
```