GBA 6430: Milestone 3: Modeling

Group 2: Giang Nguyen

Professor: Dr. Salehan

In [1]:

```
import pyspark.pandas as ps
import matplotlib.pyplot as plt
ps.set_option('plotting.backend', 'matplotlib')
import pyspark.sql.functions as F
import pyspark.ml.feature as feat
from pyspark.ml import Pipeline
import numpy as np
import pyspark.ml.regression as rg
import pyspark.ml.stat as st
import pandas as pd
import pyspark.ml.evaluation as ev
import matplotlib.pyplot as plt
student_path = 's3://giang6430/srudent/Student_preprocessing.csv'
student = ps.read_csv(student_path)
```

```
VBox()
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	
1	application_1691001566980_0002	pyspark	idle	Link (88.ec2.internal:20888/proxy/application_1691

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

```
SparkSession available as 'spark'.
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

```
/mnt1/yarn/usercache/livy/appcache/application_1691001566980_0002/contai
ner_1691001566980_0002_01_000001/pyspark.zip/pyspark/pandas/__init__.py:
50: UserWarning: 'PYARROW_IGNORE_TIMEZONE' environment variable was not
set. It is required to set this environment variable to '1' in both driv
er and executor sides if you use pyarrow>=2.0.0. pandas-on-Spark will se
t it for you but it does not work if there is a Spark context already la
unched.
/mnt1/yarn/usercache/livy/appcache/application_1691001566980_0002/contai
ner_1691001566980_0002_01_000001/pyspark.zip/pyspark/pandas/utils.py:97
5: PandasAPIOnSparkAdviceWarning: If `index_col` is not specified for `r
ead_csv`, the default index is attached which can cause additional overh
ead.
```

Preprocessing

Reminding our data, there is no null value in dataset, but we have 2.17% duplicated values so we already dropped them. Before creating model, we will check the dataset and the target variable ("Performance Score")

In [2]:
```
student.head()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

   Hours Studied  Previous Scores  Extracurricular Activities  Sleep Hou
rs  Sample Question Papers Practiced  Performance Index Performance Grou
p
0              1               40                           0
4                                  3               15.0   Low performance
1              1               40                           0
4                                  8               12.0   Low performance
2              1               40                           0
5                                  6               13.0   Low performance
3              1               40                           0
5                                  9               10.0   Low performance
4              1               40                           0
5                                  9               14.0   Low performance
```

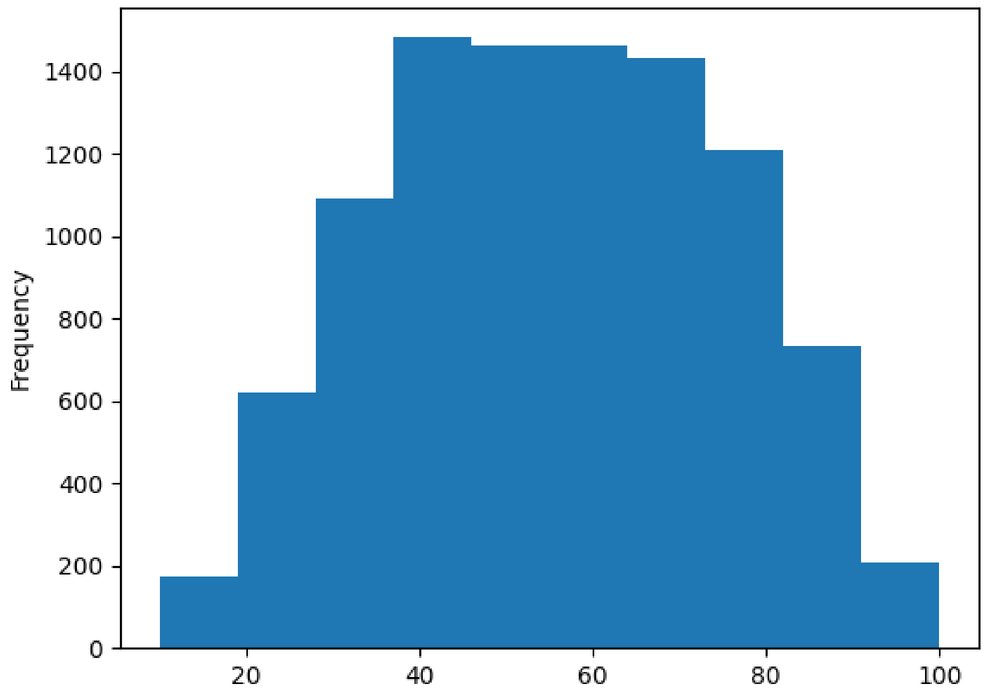
Making the histogram of "Performance index", it displays a bell-shaped distribution, the peak of the distribution occurs within the range of 44 to 60 points. This suggests that a significant number of students achieved performance scores within this range.

In [3]:

```
plt.clf()
student['Performance Index'].hist()
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```



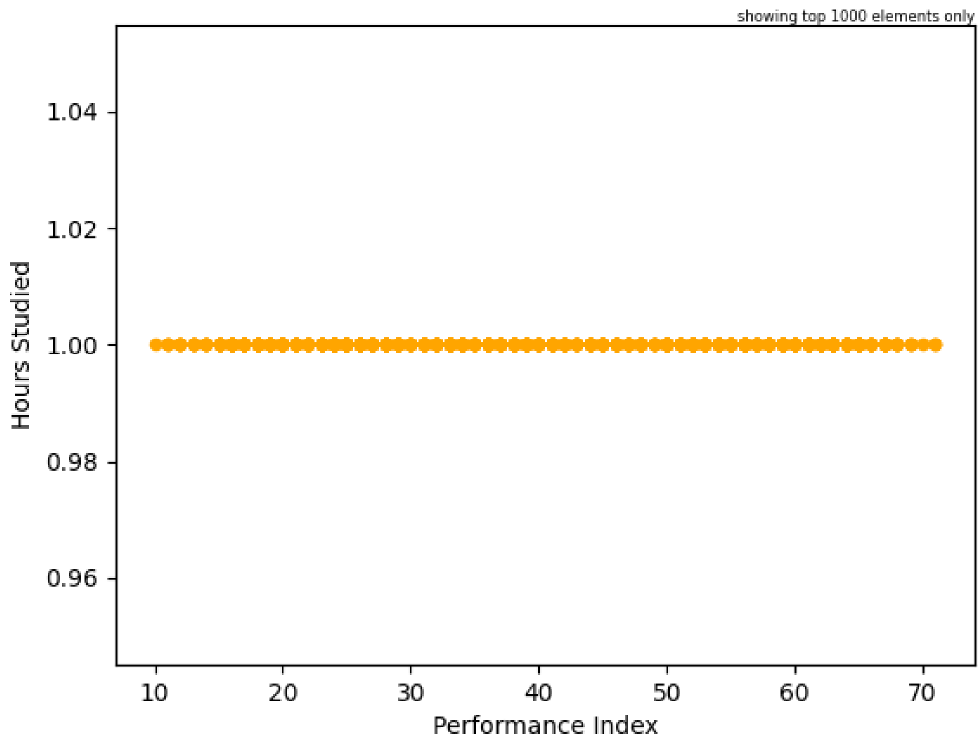
Basing on scatter plot, there is no correlation between Performance Index and Hours Studied. I will try to calculate the mean performance score per each studying hours

In [4]:

```
student.plot.scatter("Performance Index",
                      'Hours Studied', color = "orange")
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```



In [5]:

```
#Calculate average performance score
average_performance= student.groupby("Hours Studied").agg(performance_mea
average_score = average_performance.round(2)
df_sorted = average_score.reset_index().sort_values(by='Hours Studied', a

# Add a new index column and use the existing index as values
df_sorted['index'] = df_sorted.index

# Reset the dataframe index
df_sorted.set_index('index', inplace=True)
df_sorted
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

       Hours Studied  performance_mean
index
0                  1             43.98
6                  2             46.45
2                  3             49.76
3                  4             52.78
5                  5             55.50
1                  6             58.48
7                  7             60.22
4                  8             64.18
8                  9             65.81
```

In [6]:

```
#Check average score
average_score
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

               performance_mean
Hours Studied
1                         43.98
6                         58.48
3                         49.76
4                         52.78
8                         64.18
5                         55.50
2                         46.45
7                         60.22
9                         65.81
```

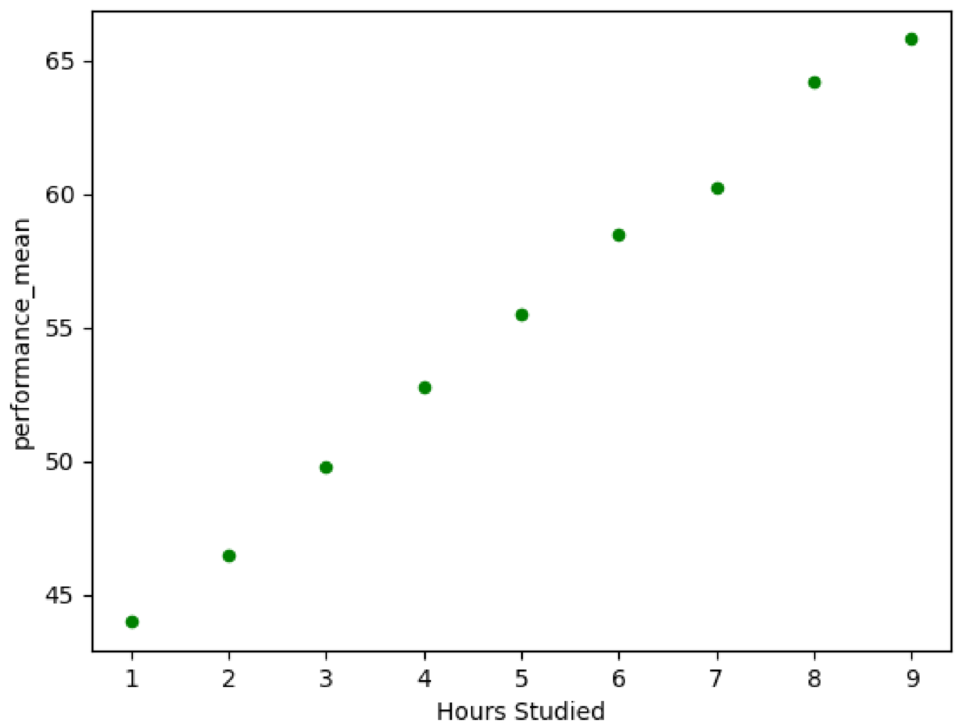
Basing on average performance metric, these plots show us the significant upward trend between hour studied and average performance score. Students study more hours, their performance scores tend to improve. It suggests that hard work, dedication, and consistent effort can lead to better academic outcomes.

In [7]:

```
df_sorted.plot.scatter('Hours Studied',
                       "performance_mean", color = "green")
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

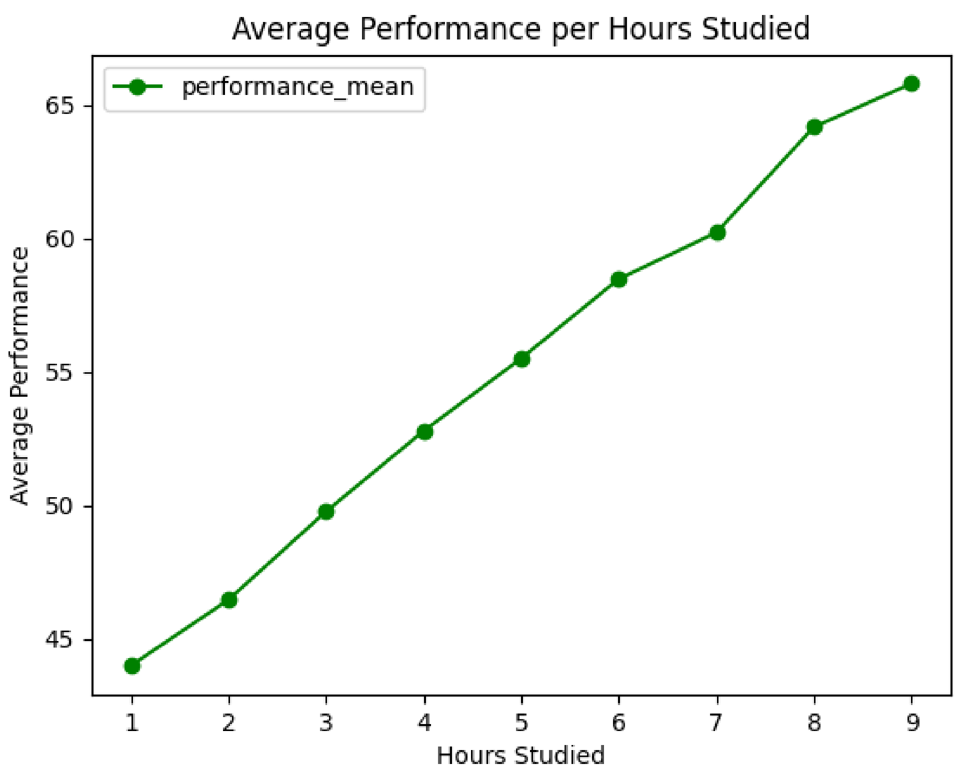


In [8]:

```
# Create a line chart using Pandas plot function
df_sorted.plot(x="Hours Studied", y='performance_mean', color='green', ma
plt.xlabel('Hours Studied')
plt.ylabel('Average Performance')
plt.title('Average Performance per Hours Studied')
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```



In [9]:

```
#try to create correlation
student.corr()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

                                  Hours Studied  Previous Scores  Extrac
urricular Activities  Sleep Hours  Sample Question Papers Practiced  Per
formance Index
Hours Studied                          1.000000        -0.010676
0.004899     0.002131                         0.015740           0.3753
32
Previous Scores                       -0.010676         1.000000
0.009534     0.007975                         0.008719           0.9151
35
Extracurricular Activities             0.004899         0.009534
1.000000    -0.024008                         0.013839           0.0260
75
Sleep Hours                            0.002131         0.007975
-0.024008     1.000000                         0.004907           0.050
352
Sample Question Papers Practiced       0.015740         0.008719
0.013839     0.004907                         1.000000           0.0434
36
Performance Index                      0.375332         0.915135
0.026075     0.050352                         0.043436           1.0000
00
```

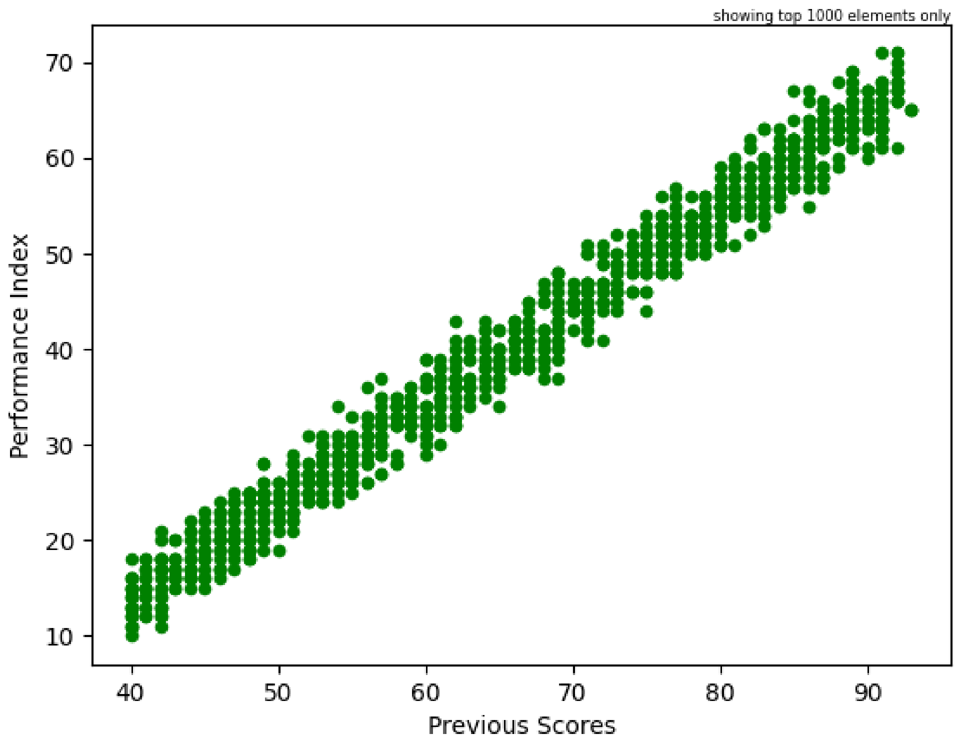
A strong positive relationship between the Performance Index and Previous Scores suggests that as the Previous Scores increase, the Performance Index also tends to increase.It implies that students who have higher previous scores are likely to achieve higher Performance Index score

In [10]:

```
student.plot.scatter('Previous Scores', 'Performance Index', color = "gre
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

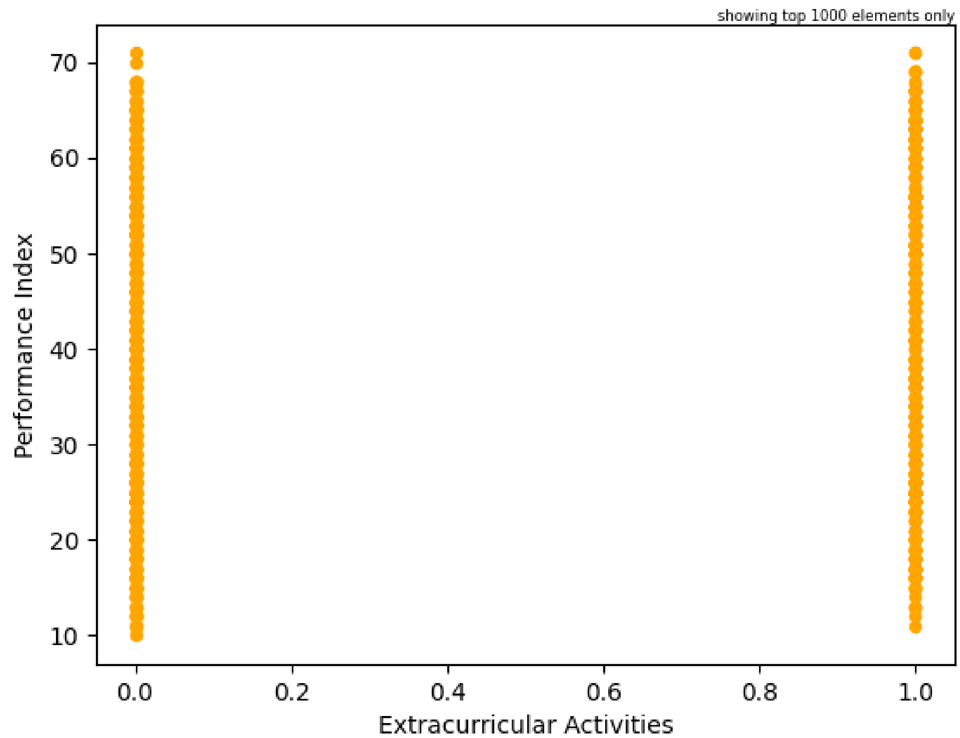


There is no relationship between Performance Index and Sample question Papers Practiced, Sleep Hour and Extracurricular Activities.

In [11]:
```
student.plot.scatter('Extracurricular Activities', 'Performance Index', c
%matplot plt
```

VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

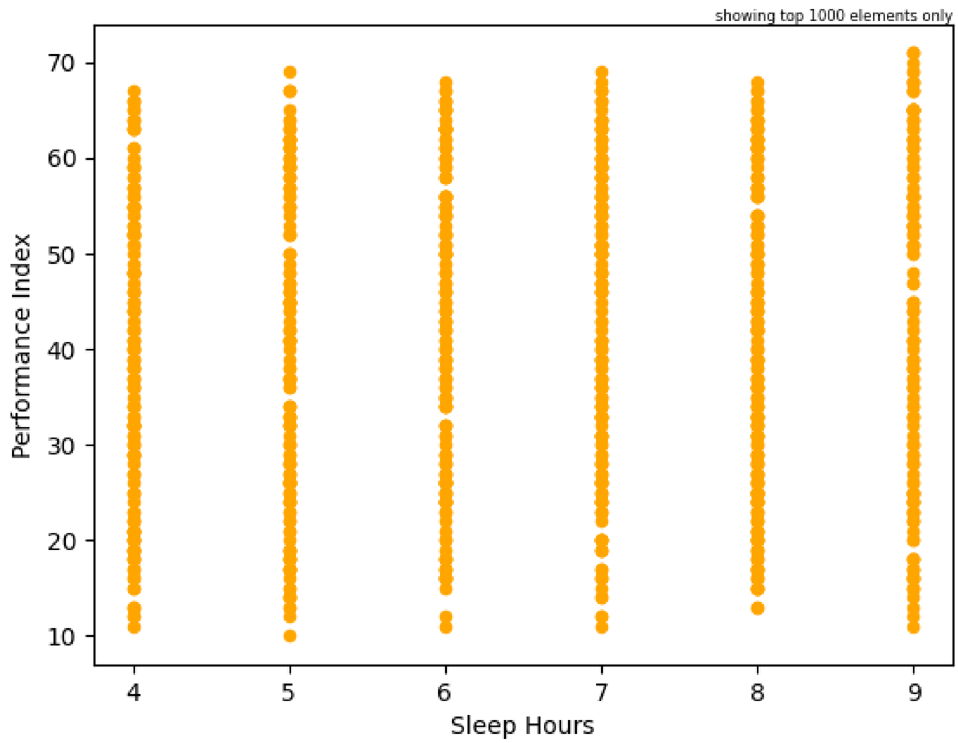


In [12]:

```
student.plot.scatter('Sleep Hours', 'Performance Index', color = "orange"
%matplot plt
```

```
VBox()
```

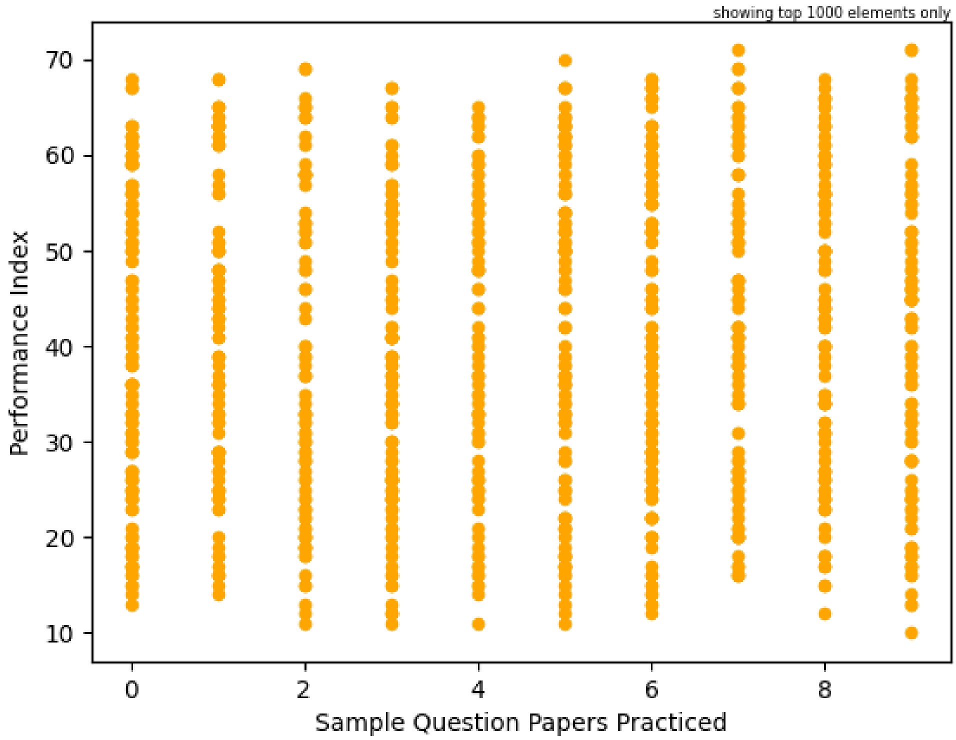
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```



In [13]:
```
student.plot.scatter('Sample Question Papers Practiced', 'Performance Ind
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```



Converting data into dataframe

In [14]:

```
student_df = student.to_spark()
student_df.show()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

```
+-------------+--------------+------------------------+----------+--
----------------------------+----------------+-----------------+
|Hours Studied|Previous Scores|Extracurricular Activities|Sleep Hours|Sa
mple Question Papers Practiced|Performance Index|Performance Group|
+-------------+--------------+------------------------+----------+--
----------------------------+----------------+-----------------+
|            1|             40|                        0|         4|
3|              15.0|  Low performance|
|            1|             40|                        0|         4|
8|              12.0|  Low performance|
|            1|             40|                        0|         5|
6|              13.0|  Low performance|
|            1|             40|                        0|         5|
9|              10.0|  Low performance|
|            1|             40|                        0|         5|
9|              14.0|  Low performance|
|            1|             40|                        0|         6|
0|              15.0|  Low performance|
|            1|             40|                        0|         6|
3|              12.0|  Low performance|
|            1|             40|                        0|         6|
6|              16.0|  Low performance|
|            1|             40|                        0|         7|
4|              11.0|  Low performance|
|            1|             40|                        0|         7|
4|              14.0|  Low performance|
|            1|             40|                        0|         8|
5|              16.0|  Low performance|
|            1|             40|                        0|         9|
2|              11.0|  Low performance|
|            1|             40|                        0|         9|
6|              13.0|  Low performance|
|            1|             40|                        0|         9|
6|              14.0|  Low performance|
|            1|             40|                        0|         9|
7|              16.0|  Low performance|
|            1|             40|                        1|         4|
2|              13.0|  Low performance|
|            1|             40|                        1|         4|
3|              13.0|  Low performance|
|            1|             40|                        1|         6|
5|              11.0|  Low performance|
|            1|             40|                        1|         7|
6|              12.0|  Low performance|
|            1|             40|                        1|         8|
8|              15.0|  Low performance|
+-------------+--------------+------------------------+----------+--
----------------------------+----------------+-----------------+
only showing top 20 rows

/mnt1/yarn/usercache/livy/appcache/application_1691001566980_0002/contai
ner_1691001566980_0002_01_000001/pyspark.zip/pyspark/pandas/utils.py:97
5: PandasAPIOnSparkAdviceWarning: If `index_col` is not specified for `t
o_spark`, the existing index is lost when converting to Spark DataFrame.
  warnings.warn(message, PandasAPIOnSparkAdviceWarning)
```

In this project, we will focus on predicting performance score, so we will go with Linear Regression and drop Performance group

In [15]:

```
#We will focus on linear regression
spark_df = student_df.drop("Performance Group")
spark_df.show(5)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

+-------------+--------------+--------------------------+-----------+--
-----------------------------+-----------------+
|Hours Studied|Previous Scores|Extracurricular Activities|Sleep Hours|Sa
mple Question Papers Practiced|Performance Index|
+-------------+--------------+--------------------------+-----------+--
-----------------------------+-----------------+
|            1|            40|                         0|          4|
3|             15.0|
|            1|            40|                         0|          4|
8|             12.0|
|            1|            40|                         0|          5|
6|             13.0|
|            1|            40|                         0|          5|
9|             10.0|
|            1|            40|                         0|          5|
9|             14.0|
+-------------+--------------+--------------------------+-----------+--
-----------------------------+-----------------+
only showing top 5 rows
```

In [16]:

```
student[features].corr()
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

An error was encountered:
name 'features' is not defined
Traceback (most recent call last):
NameError: name 'features' is not defined
```

Correlation

To figure out the correlation, we try to create the correlation matrix. First of all, we convert data in to vector. Then, we calculate the correlation scores and put them in dataframe

In [17]:

```python
#Converting data to vector
features = feat.VectorAssembler(
    inputCols=list(spark_df.columns)
    , outputCol='features'
)
#Creating the correlation matrix score
corr = st.Correlation.corr(
    features.transform(spark_df),
    'features',
    'pearson'
)

print(str(corr.collect()[0][0]))
corr_pd = corr.toPandas()
output_np = np.array(corr_pd.iloc[0, 0].values).reshape(
    (corr_pd.iloc[0, 0].numRows, corr_pd.iloc[0, 0].numCols))

corr_pd = pd.DataFrame(output_np, columns=spark_df.columns)
corr_pd.index = spark_df.columns
corr_pd = ps.from_pandas(corr_pd)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

DenseMatrix([[ 1.        , -0.01067582,  0.00489909,  0.00213102,  0.015
73978,
               0.37533203],
             [-0.01067582,  1.        ,  0.00953394,  0.00797488,  0.008
71942,
               0.91513508],
             [ 0.00489909,  0.00953394,  1.        , -0.02400822,  0.013
83914,
               0.02607459],
             [ 0.00213102,  0.00797488, -0.02400822,  1.        ,  0.004
90732,
               0.05035247],
             [ 0.01573978,  0.00871942,  0.01383914,  0.00490732,  1.
,
               0.04343571],
             [ 0.37533203,  0.91513508,  0.02607459,  0.05035247,  0.043
43571,
               1.        ]])
```

In [18]:
```
print(corr_pd)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

                                  Hours Studied  Previous Scores  Extrac
urricular Activities  Sleep Hours  Sample Question Papers Practiced  Per
formance Index
Hours Studied                          1.000000        -0.010676
0.004899     0.002131                         0.015740          0.3753
32
Previous Scores                       -0.010676         1.000000
0.009534     0.007975                         0.008719          0.9151
35
Extracurricular Activities             0.004899         0.009534
1.000000    -0.024008                         0.013839          0.0260
75
Sleep Hours                            0.002131         0.007975
-0.024008     1.000000                         0.004907          0.050
352
Sample Question Papers Practiced       0.015740         0.008719
0.013839     0.004907                         1.000000          0.0434
36
Performance Index                      0.375332         0.915135
0.026075     0.050352                         0.043436          1.0000
00
```

We have a weak positive correlation of 0.37 between the Perforamnce Index and Hours Studied. This means that students who study more hours tend to have slightly better Performance Index scores, but other factors may also be influencing their performance

We have a strong correlation of 0.95 between the Performance Index and the Previous Score (student test score). This means that students who achieve higher test scores are likely to have better performance scores.

We do not have any correlation between Performance with the rest of variables

In [19]:
```
#Save csv file to S3 bucket
# corr_pd.to_spark(index_col='index').repartition(1).pandas_api().to_csv(
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

Regression

Regard normalizing data before building predictive model, we decide not to normalized data since the amount of dataset is not too big and all the features are in similar scale range (0-100).

1. Transformer data

In [20]:

```
vectorAssembler = feat.VectorAssembler(
    inputCols=[x for x in spark_df.columns if x != 'Performance Index']
    , outputCol='features'
)
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

We transform multiple columns in to single vectosr named "features", which contains "Hours Studied","Extra Activities","Previous Scores","Sleep Hours" and "Sample Questions".

2. Create Linear Regression object and pipeline

In [21]:
```
#create a linear regression object and fit to dataset
lr_obj = rg.LinearRegression(
    labelCol='Performance Index',
    maxIter=10
    , regParam=0.01
    , elasticNetParam=0.7)

#examine model coefficients
pip = Pipeline(stages=[vectorAssembler, lr_obj])

#run the pipeline
pModel = pip.fit(spark_df)
#get the trained model from the pipeline
lr_model = pModel.stages[-1]
#examine model coefficients
lr_model.coefficients
summary = lr_model.summary

print(
    summary.r2
    , summary.rootMeanSquaredError
    , summary.meanAbsoluteError
)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

0.9886805456497294 2.0435524407269923 1.6231468408447451
```

We achieved an R-squared value of 98.87%, a root mean square error (RMSE) of 2.04, and a mean absolute error (MAE) of 1.62.

Regard R-square score, the model explains 98.87% of the variance in the target variable, which is a very high level of explanation.

Regard RMSE score, it measures the average difference between the actual values and the predicted values. A lower RMSE indicates better accuracy of the model. In our prediction, an RMSE of 2.04 suggests that, predicted values differ from the actual values by approximately 2.04 on average.

Regard MAE score, it measures the average absolute difference between the actual and predicted values. Similarly RMSE, a lower MAE indicates better model accuracy, so MAE of 1.62 means the absolute difference between predicted and actual values is approximately 1.62 on average

In [22]:

```
coefficients = lr_model.coefficients
intercept = lr_model.intercept

print("Coefficients: ", coefficients)
print("Intercept: {:.3f}".format(intercept))
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

Coefficients:  [2.8495999366686067,1.0177619895493197,0.602704679826610
8,0.47609958529775187,0.19155839442032715]
Intercept: -33.969
```

We calculate the coefficients of each variables as follows:

Performance equation = 2.86 * Hour Studied + 1.02 * Previous Scores + 0.61 * Activities+ 0.4 * sleep + 0.19 * Sample Question Papers Practiced - 34.08

The coefficients indicate how each predictor variable contributes to the overall performance, and the intercept term (-34.08) represents the expected performance when all predictor variables are zero.

Hours Studied: it has a coefficient of 2.86, which means that, on average, for every additional hour studied, the predicted performance is expected to increase by 2.86 score units

Previous Scores: With every one-unit increase in previous test scores, the predicted performance is expected to increase by 1.02 score units

Extra Activities: with every one-unit increase in extra activities, the predicted performance is expected to increase by 0.61 units, while keeping other variables constant.

Sleep Hours: with every additional hour of sleep, the predicted performance is expected to increase by 0.4 score

Sample Question Papers Practiced: with every additional practice of a sample question paper, the predicted performance is expected to increase by 0.19 score, holding other variables constant.

In [23]:

```
#In this cell, we will try to get predictions from the model
(
    pModel.transform(spark_df)
    .pandas_api()
    [["features",'Performance Index', 'prediction']]
    .head(5)
)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

                        features  Performance Index  prediction
0  [1.0, 40.0, 0.0, 4.0, 3.0]               15.0   12.069843
1  [1.0, 40.0, 0.0, 4.0, 8.0]               12.0   13.027635
2  [1.0, 40.0, 0.0, 5.0, 6.0]               13.0   13.120618
3  [1.0, 40.0, 0.0, 5.0, 9.0]               10.0   13.695293
4  [1.0, 40.0, 0.0, 5.0, 9.0]               14.0   13.695293
```

3. Gradient Boosting Regression

After running Linear Regression, we willtry to run Gradient Boosting Regression to see better predictive performance.

In [24]:

```
gbt_obj = rg.GBTRegressor(
    labelCol='Performance Index'
    , minInstancesPerNode=10
    , minInfoGain=0.1
)

pip = Pipeline(stages=[vectorAssembler, gbt_obj])

results = (
    pip
    .fit(spark_df)
    .transform(spark_df)
    .select('Performance Index', 'prediction')
)

evaluator = ev.RegressionEvaluator(labelCol='Performance Index')
r2 = evaluator.evaluate(results, {evaluator.metricName: 'r2'})
rmse = evaluator.evaluate(results, {evaluator.metricName: 'rmse'})
mae = evaluator.evaluate(results, {evaluator.metricName: 'mae'})
print("RMSE: ", rmse)
print("MAE: ", mae)
print("R-squared: ", r2)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

RMSE:  2.83521508326469
MAE:  2.2607333069394597
R-squared:  0.9782115698822962
```

Overall, the performance metrics of Gradient Boosting Regression model are good with R square 0.978. The low RMSE (2.83) and MAE (2.26) values and the high R-squared value indicate that the model is performing well and accurately predicting the target variable. It seems that the Gradient Boosting Regression model is a strong fit (shown as picture belows).

In [25]:

```
x_ax = range(0, results.count())
y_pred=results.select("prediction").collect()
y_orig=results.select("Performance Index").collect()

plt.plot(x_ax, y_orig, label="original")
plt.plot(x_ax, y_pred, label="predicted")
plt.title("Perfomance score test and predicted data")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend(loc='best',fancybox= False, shadow= False)
plt.grid(False)
%matplot plt
```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

4. Tune hyperparameters

In [26]:

```
lr_tune = rg.LinearRegression(
    labelCol='Performance Index',
    maxIter=10
    , regParam=0.3
    , elasticNetParam=0.9)
#examine model coefficients
pip2 = Pipeline(stages=[vectorAssembler, lr_tune])

#run the pipeline
pModel2 = pip2.fit(spark_df)
#get the trained model from the pipeline
lr_model2 = pModel2.stages[-1]
#examine model coefficients
lr_model2.coefficients
summary2 = lr_model2.summary

print(
    summary2.r2
    , summary2.rootMeanSquaredError
    , summary2.meanAbsoluteError
)
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

0.9876465511948898 2.134848941998442 1.6984185802257774
```

If we try to tune hyperparameters for our Linear Regression model with regularization parameter (regParam = 0.3) and the elastic net mixing parameter (elasticNetparam = 0.9). The results we achieved are R-square 0.987, RMSE 2.13 and MAE 1.7. These results indicate that the tuned Linear Regression model is performing very well and has high accuracy in predicting the target variable

5. Feature Importance

In [27]:

```
#select top 5 features, store in a new column named selected
selector = feat.UnivariateFeatureSelector(
    labelCol='Performance Index'
    , outputCol='selected'
    , selectionMode = 'numTopFeatures'
    ).setFeatureType("categorical"
    ).setLabelType("categorical"
    ).setSelectionThreshold(5) #select top 5 features
#Create pipeline
pipeline_sel = Pipeline(stages=[vectorAssembler, selector])

model = (
    pipeline_sel
    .fit(spark_df)
    .transform(spark_df)
)

#print selected features
model.schema['selected'].metadata
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

{'ml_attr': {'attrs': {'numeric': [{'idx': 0, 'name': 'Hours Studied'},
{'idx': 1, 'name': 'Previous Scores'}, {'idx': 2, 'name': 'Extracurricul
ar Activities'}, {'idx': 3, 'name': 'Sleep Hours'}, {'idx': 4, 'name':
'Sample Question Papers Practiced'}]}, 'num_attrs': 5}}
```

In [28]:

```
#display selected features as a dataframe
pd.DataFrame(model.schema['selected'].metadata['ml_attr']['attrs']['numer
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…
```

	idx	name
0	0	Hours Studied
1	1	Previous Scores
2	2	Extracurricular Activities
3	3	Sleep Hours
4	4	Sample Question Papers Practiced

In [29]:

```
feature_importance = sorted(list(zip(spark_df.columns[:-1], map(abs, coef

print("Feature Importance:")
for feature, importance in feature_importance:
    print("  {}: {:.3f}".format(feature, importance))
```

```
VBox()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layo
ut=Layout(height='25px', width='50%'),…

Feature Importance:
  Hours Studied: 2.850
  Previous Scores: 1.018
  Extracurricular Activities: 0.603
  Sleep Hours: 0.476
  Sample Question Papers Practiced: 0.192
```

The feature importance values represent the relative importance of each predictor variable in the Linear Regression model. These values indicate how much each predictor contributes to the predictions made by the model.

"Hours Studied" and "Previous Scores" have the most significant impact, followed by "Extracurricular Activities," "Sleep Hours," and "Sample Question Papers Practiced" in descending order of importance. These insights can be valuable for understanding the factors that contribute most to the model's predictions and can help make informed decisions and recommendations based on the model's results.

Making recomendations:

Encourage students to allocate more time to study ("Hours Studied") to improve their academic performance.

Emphasize the importance of building on previous achievements and striving to improve "Previous Scores.

Encourage students to participate in "Extracurricular Activities" as they can contribute positively to the predicted outcome. It can enhance skills, well-being, and personal development.

Emphasize the importance of adequate sleep for overall health and well-being, even though its impact on the predicted outcome is moderate (Sleep Hours)

Encourage students to practice sample question papers as a valuable study strategy, even though it has the least importance.

In [ ]:

```
```