In [3]:

import pyspark

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

df = spark.read.csv("Absenteeism\_at\_work.csv", header=True, inferSchema=True)

df.printSchema()

root

|-- ID: integer (nullable = true)

|-- Reason for absence: string (nullable = true)

|-- Month of absence: string (nullable = true)

|-- Day of the week: string (nullable = true)

|-- Seasons: string (nullable = true)

|-- Transportation expense: integer (nullable = true)

|-- Distance from Residence to Work: integer (nullable = true)

|-- Service time: integer (nullable = true)

|-- Age: integer (nullable = true)

|-- Work load Average/day : double (nullable = true)

|-- Hit target: integer (nullable = true)

|-- Disciplinary failure: boolean (nullable = true)

|-- Education: string (nullable = true)

|-- Children: integer (nullable = true)

|-- Social drinker: boolean (nullable = true)

|-- Social smoker: boolean (nullable = true)

|-- Pet: integer (nullable = true)

|-- Weight: integer (nullable = true)

|-- Height: integer (nullable = true)

|-- Body mass index: integer (nullable = true)

|-- Absenteeism time in hours: integer (nullable = true)

In [5]:

df.count()

Out[5]:

740

In [6]:

df.na.drop().count()

Out[6]:

740

In [4]:

from pyspark.sql.functions import col

cat\_df = df.select(col('Reason for Absence'),

col('Day of the week'),

col('Seasons'),

col('Disciplinary failure'),

col('Education'),

col('Social drinker'),

col('Social smoker')

)

cat\_df.show()

+------------------+---------------+-------+--------------------+-----------+--------------+-------------+

|Reason for Absence|Day of the week|Seasons|Disciplinary failure| Education|Social drinker|Social smoker|

+------------------+---------------+-------+--------------------+-----------+--------------+-------------+

| XXVI| TUES| Summer| false|High School| true| false|

| 0| TUES| Summer| true|High School| true| false|

| XXIII| WED| Summer| false|High School| true| false|

| VII| THURS| Summer| false|High School| true| true|

| XXIII| THURS| Summer| false|High School| true| false|

| XXIII| FRI| Summer| false|High School| true| false|

| XXII| FRI| Summer| false|High School| true| false|

| XXIII| FRI| Summer| false|High School| true| false|

| XIX| MON| Summer| false|High School| true| false|

| XXII| MON| Summer| false| Postgrad| false| false|

| I| MON| Summer| false|High School| true| false|

| I| TUES| Summer| false|High School| true| false|

| XI| WED| Summer| false|High School| true| false|

| XI| WED| Summer| false|High School| true| false|

| XXIII| WED| Summer| false|High School| true| false|

| XIV| FRI| Summer| false|High School| true| false|

| XXIII| FRI| Summer| false|High School| true| false|

| XXI| MON| Summer| false|High School| true| false|

| XI| THURS| Summer| false|High School| false| false|

| XXIII| WED| Summer| false|High School| false| false|

+------------------+---------------+-------+--------------------+-----------+--------------+-------------+

only showing top 20 rows

cat\_df.describe().toPandas()

In [8]:

num\_df = df.select(col('Transportation expense'),

col('Distance from Residence to Work'),

col('Service Time'),

col('Age'),

col('Work load Average/Day '),

col('Hit target'),

col('Children'),

col('Pet'),

col('Weight'),

col('Height'),

col('Body Mass Index'),

col('Absenteeism time in hours')

)

num\_df.show()

+----------------------+-------------------------------+------------+---+----------------------+----------+--------+---+------+------+---------------+-------------------------+

|Transportation expense|Distance from Residence to Work|Service Time|Age|Work load Average/Day |Hit target|Children|Pet|Weight|Height|Body Mass Index|Absenteeism time in hours|

+----------------------+-------------------------------+------------+---+----------------------+----------+--------+---+------+------+---------------+-------------------------+

| 289| 36| 13| 33| 239.554| 97| 2| 1| 90| 172| 30| 4|

| 118| 13| 18| 50| 239.554| 97| 1| 0| 98| 178| 31| 0|

| 179| 51| 18| 38| 239.554| 97| 0| 0| 89| 170| 31| 2|

| 279| 5| 14| 39| 239.554| 97| 2| 0| 68| 168| 24| 4|

| 289| 36| 13| 33| 239.554| 97| 2| 1| 90| 172| 30| 2|

| 179| 51| 18| 38| 239.554| 97| 0| 0| 89| 170| 31| 2|

| 361| 52| 3| 28| 239.554| 97| 1| 4| 80| 172| 27| 8|

| 260| 50| 11| 36| 239.554| 97| 4| 0| 65| 168| 23| 4|

| 155| 12| 14| 34| 239.554| 97| 2| 0| 95| 196| 25| 40|

| 235| 11| 14| 37| 239.554| 97| 1| 1| 88| 172| 29| 8|

| 260| 50| 11| 36| 239.554| 97| 4| 0| 65| 168| 23| 8|

| 260| 50| 11| 36| 239.554| 97| 4| 0| 65| 168| 23| 8|

| 260| 50| 11| 36| 239.554| 97| 4| 0| 65| 168| 23| 8|

| 179| 51| 18| 38| 239.554| 97| 0| 0| 89| 170| 31| 1|

| 179| 51| 18| 38| 239.554| 97| 0| 0| 89| 170| 31| 4|

| 246| 25| 16| 41| 239.554| 97| 0| 0| 67| 170| 23| 8|

| 179| 51| 18| 38| 239.554| 97| 0| 0| 89| 170| 31| 2|

| 179| 51| 18| 38| 239.554| 97| 0| 0| 89| 170| 31| 8|

| 189| 29| 13| 33| 239.554| 97| 2| 2| 69| 167| 25| 8|

| 248| 25| 14| 47| 205.917| 92| 2| 1| 86| 165| 32| 2|

+----------------------+-------------------------------+------------+---+----------------------+----------+--------+---+------+------+---------------+-------------------------+

only showing top 20 rows

In [10]:

num\_df.describe().toPandas()

Out[10]:

|  | **summary** | **Transportation expense** | **Distance from Residence to Work** | **Service Time** | **Age** | **Work load Average/Day** | **Hit target** | **Children** | **Pet** | **Weight** | **Height** | **Body Mass Index** | **Absenteeism time in hours** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | count | 740 | 740 | 740 | 740 | 740 | 740 | 740 | 740 | 740 | 740 | 740 | 740 |
| **1** | mean | 221.32972972972973 | 29.63108108108108 | 12.554054054054054 | 36.45 | 271.49023513513544 | 94.58783783783784 | 1.018918918918919 | 0.745945945945946 | 79.03513513513514 | 172.11486486486487 | 26.677027027027027 | 6.924324324324324 |
| **2** | stddev | 66.95222324531973 | 14.836788436739143 | 4.384873407621148 | 6.478772457611868 | 39.058116188144055 | 3.779313134418015 | 1.0984890195302817 | 1.3182582913258336 | 12.883210507177214 | 6.034994530267662 | 4.285452223167274 | 13.330998100978196 |
| **3** | min | 118 | 5 | 1 | 27 | 205.917 | 81 | 0 | 0 | 56 | 163 | 19 | 0 |
| **4** | max | 388 | 52 | 29 | 58 | 378.884 | 100 | 4 | 8 | 108 | 196 | 38 | 120 |

In [19]:

# Look at frequency of Boolean variables

cat\_df.groupBy('Social Smoker').count().show()

+-------------+-----+

|Social Smoker|count|

+-------------+-----+

| true| 54|

| false| 686|

+-------------+-----+

In [20]:

cat\_df.groupBy('Social drinker').count().show()

+--------------+-----+

|Social drinker|count|

+--------------+-----+

| true| 420|

| false| 320|

+--------------+-----+

In [26]:

cat\_df.groupBy('Reason for Absence').count().show(28)

+------------------+-----+

|Reason for Absence|count|

+------------------+-----+

| XVII| 1|

| VII| 15|

| XXVII| 69|

| XVI| 3|

| XIII| 55|

| 0| 43|

| XIV| 19|

| XXV| 31|

| XXIII| 149|

| III| 1|

| IV| 2|

| XXVI| 33|

| V| 3|

| VI| 8|

| IX| 4|

| XV| 2|

| X| 25|

| XIX| 40|

| II| 1|

| XXI| 6|

| XVIII| 21|

| XXIV| 3|

| XXII| 38|

| I| 16|

| XI| 26|

| VIII| 6|

| XII| 8|

| XXVIII| 112|

+------------------+-----+

In [27]:

cat\_df.groupBy('Seasons').count().show()

+-------+-----+

|Seasons|count|

+-------+-----+

| Spring| 195|

| Summer| 170|

| Autumn| 192|

| Winter| 183|

+-------+-----+

In [28]:

cat\_df.groupBy('Day of the Week').count().show()

+---------------+-----+

|Day of the Week|count|

+---------------+-----+

| TUES| 154|

| MON| 161|

| THURS| 125|

| WED| 156|

| FRI| 144|

+---------------+-----+

In [32]:

cat\_df.groupBy('Education').count().show()

+-----------+-----+

| Education|count|

+-----------+-----+

|High School| 611|

| Postgrad| 79|

| Doctorate| 4|

| Graduate| 46|

+-----------+-----+

In [35]:

cat\_df.groupBy('Disciplinary failure').count().show()

+--------------------+-----+

|Disciplinary failure|count|

+--------------------+-----+

| true| 40|

| false| 700|

+--------------------+-----+

In [30]:

# https://stackoverflow.com/questions/51831874/how-to-get-correlation-matrix-values-pyspark

from pyspark.ml.linalg import DenseMatrix, Vectors

from pyspark.ml.stat import Correlation

from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols=df.columns, outputCol="features", handleInvalid='keep')

dataframe = assembler.transform(df).select("features")

dataframe.show()

---------------------------------------------------------------------------

IllegalArgumentException Traceback (most recent call last)

C:\Users\SARAHW~1\AppData\Local\Temp/ipykernel\_10488/1317897778.py in <module>

4 from pyspark.ml.feature import VectorAssembler

5 assembler = VectorAssembler(inputCols=df.columns, outputCol="features", handleInvalid='keep')

----> 6 dataframe = assembler.transform(df).select("features")

7 dataframe.show()

~\anaconda3\lib\site-packages\pyspark\ml\base.py in transform(self, dataset, params)

215 return self.copy(params).\_transform(dataset)

216 else:

--> 217 return self.\_transform(dataset)

218 else:

219 raise TypeError("Params must be a param map but got %s." % type(params))

~\anaconda3\lib\site-packages\pyspark\ml\wrapper.py in \_transform(self, dataset)

348 def \_transform(self, dataset):

349 self.\_transfer\_params\_to\_java()

--> 350 return DataFrame(self.\_java\_obj.transform(dataset.\_jdf), dataset.sql\_ctx)

351

352

~\anaconda3\lib\site-packages\py4j\java\_gateway.py in \_\_call\_\_(self, \*args)

1319

1320 answer = self.gateway\_client.send\_command(command)

-> 1321 return\_value = get\_return\_value(

1322 answer, self.gateway\_client, self.target\_id, self.name)

1323

~\anaconda3\lib\site-packages\pyspark\sql\utils.py in deco(\*a, \*\*kw)

115 # Hide where the exception came from that shows a non-Pythonic

116 # JVM exception message.

--> 117 raise converted from None

118 else:

119 raise

IllegalArgumentException: Data type string of column Reason for absence is not supported.

Data type string of column Month of absence is not supported.

Data type string of column Day of the week is not supported.

Data type string of column Seasons is not supported.

Data type string of column Education is not supported.

In [36]:

# String variables not supported in correlations. Best to remove

new\_df = df.select(col('Transportation expense'),

col('Distance from Residence to Work'),

col('Service Time'),

col('Age'),

col('Work load Average/Day '),

col('Hit target'),

col('Children'),

col('Pet'),

col('Weight'),

col('Height'),

col('Body Mass Index'),

col('Absenteeism time in hours'), col('Social Smoker'), col('Social Drinker'), col('Disciplinary failure'),

)

In [37]:

assembler = VectorAssembler(inputCols=new\_df.columns, outputCol="features", handleInvalid='keep')

dataframe = assembler.transform(new\_df).select("features")

dataframe.show()

+--------------------+

| features|

+--------------------+

|[289.0,36.0,13.0,...|

|[118.0,13.0,18.0,...|

|[179.0,51.0,18.0,...|

|[279.0,5.0,14.0,3...|

|[289.0,36.0,13.0,...|

|[179.0,51.0,18.0,...|

|[361.0,52.0,3.0,2...|

|[260.0,50.0,11.0,...|

|[155.0,12.0,14.0,...|

|[235.0,11.0,14.0,...|

|[260.0,50.0,11.0,...|

|[260.0,50.0,11.0,...|

|[260.0,50.0,11.0,...|

|[179.0,51.0,18.0,...|

|[179.0,51.0,18.0,...|

|[246.0,25.0,16.0,...|

|[179.0,51.0,18.0,...|

|[179.0,51.0,18.0,...|

|[189.0,29.0,13.0,...|

|[248.0,25.0,14.0,...|

+--------------------+

only showing top 20 rows

In [39]:

correlation = Correlation.corr(dataframe,"features","pearson").collect()[0][0]

print(str(correlation))

DenseMatrix([[ 1.00000000e+00, 2.62183111e-01, -3.49887036e-01,

-2.27542434e-01, 5.43806459e-03, -8.01930398e-02,

3.83001191e-01, 4.00080301e-01, -2.07434941e-01,

-1.94495956e-01, -1.36516573e-01, 2.75846310e-02,

4.43561444e-02, 1.45117454e-01, 1.09221690e-01],

[ 2.62183111e-01, 1.00000000e+00, 1.31730304e-01,

-1.45886369e-01, -6.86769583e-02, -1.38646253e-02,

5.42303931e-02, 2.05940581e-01, -4.78590935e-02,

-3.53372180e-01, 1.13771638e-01, -8.83628219e-02,

-7.53687916e-02, 4.52195702e-01, -5.65270951e-02],

[-3.49887036e-01, 1.31730304e-01, 1.00000000e+00,

6.70978917e-01, -6.68491033e-04, -7.84003475e-03,

-4.71284116e-02, -4.40300667e-01, 4.55974804e-01,

-5.31345133e-02, 4.99717950e-01, 1.90292614e-02,

7.24243087e-02, 3.53140609e-01, -2.21160260e-04],

[-2.27542434e-01, -1.45886369e-01, 6.70978917e-01,

1.00000000e+00, -3.94251764e-02, -3.92243142e-02,

5.69841215e-02, -2.31225999e-01, 4.18730459e-01,

-6.29965830e-02, 4.70688017e-01, 6.57597013e-02,

1.21738391e-01, 2.13182620e-01, 1.04303851e-01],

[ 5.43806459e-03, -6.86769583e-02, -6.68491033e-04,

-3.94251764e-02, 1.00000000e+00, -8.94449560e-02,

2.78202361e-02, 7.11419762e-03, -3.85215697e-02,

1.03314799e-01, -9.07092809e-02, 2.47488997e-02,

3.09683237e-02, -3.37126191e-02, 2.90257512e-02],

[-8.01930398e-02, -1.38646253e-02, -7.84003475e-03,

-3.92243142e-02, -8.94449560e-02, 1.00000000e+00,

-1.40906000e-02, 7.20127601e-03, -4.49474247e-02,

9.32669294e-02, -8.89394521e-02, 2.66950646e-02,

5.12538947e-02, -1.02479904e-01, -1.47970834e-01],

[ 3.83001191e-01, 5.42303931e-02, -4.71284116e-02,

5.69841215e-02, 2.78202361e-02, -1.40906000e-02,

1.00000000e+00, 1.08917279e-01, -1.39552437e-01,

-1.42083220e-02, -1.44150249e-01, 1.13756496e-01,

1.56087556e-01, 2.06375968e-01, 7.20963471e-02],

[ 4.00080301e-01, 2.05940581e-01, -4.40300667e-01,

-2.31225999e-01, 7.11419762e-03, 7.20127601e-03,

1.08917279e-01, 1.00000000e+00, -1.03770410e-01,

-1.03143350e-01, -7.61029459e-02, -2.82765891e-02,

1.05378757e-01, -1.22780216e-01, 1.88813529e-02],

[-2.07434941e-01, -4.78590935e-02, 4.55974804e-01,

4.18730459e-01, -3.85215697e-02, -4.49474247e-02,

-1.39552437e-01, -1.03770410e-01, 1.00000000e+00,

3.06801835e-01, 9.04116901e-01, 1.57891765e-02,

-1.98511210e-01, 3.78663607e-01, 7.22245804e-02],

[-1.94495956e-01, -3.53372180e-01, -5.31345133e-02,

-6.29965830e-02, 1.03314799e-01, 9.32669294e-02,

-1.42083220e-02, -1.03143350e-01, 3.06801835e-01,

1.00000000e+00, -1.21048777e-01, 1.44420476e-01,

3.27139076e-03, 1.69951078e-01, -1.04983751e-02],

[-1.36516573e-01, 1.13771638e-01, 4.99717950e-01,

4.70688017e-01, -9.07092809e-02, -8.89394521e-02,

-1.44150249e-01, -7.61029459e-02, 9.04116901e-01,

-1.21048777e-01, 1.00000000e+00, -4.97194791e-02,

-1.96006483e-01, 3.23977794e-01, 7.94281798e-02],

[ 2.75846310e-02, -8.83628219e-02, 1.90292614e-02,

6.57597013e-02, 2.47488997e-02, 2.66950646e-02,

1.13756496e-01, -2.82765891e-02, 1.57891765e-02,

1.44420476e-01, -4.97194791e-02, 1.00000000e+00,

-8.93642314e-03, 6.50673428e-02, -1.24247984e-01],

[ 4.43561444e-02, -7.53687916e-02, 7.24243087e-02,

1.21738391e-01, 3.09683237e-02, 5.12538947e-02,

1.56087556e-01, 1.05378757e-01, -1.98511210e-01,

3.27139076e-03, -1.96006483e-01, -8.93642314e-03,

1.00000000e+00, -1.11678005e-01, 1.16748119e-01],

[ 1.45117454e-01, 4.52195702e-01, 3.53140609e-01,

2.13182620e-01, -3.37126191e-02, -1.02479904e-01,

2.06375968e-01, -1.22780216e-01, 3.78663607e-01,

1.69951078e-01, 3.23977794e-01, 6.50673428e-02,

-1.11678005e-01, 1.00000000e+00, 5.18380278e-02],

[ 1.09221690e-01, -5.65270951e-02, -2.21160260e-04,

1.04303851e-01, 2.90257512e-02, -1.47970834e-01,

7.20963471e-02, 1.88813529e-02, 7.22245804e-02,

-1.04983751e-02, 7.94281798e-02, -1.24247984e-01,

1.16748119e-01, 5.18380278e-02, 1.00000000e+00]])

In [41]:

rows = correlation.toArray().tolist()

print(str(rows))

correlationdf = spark.createDataFrame(rows, new\_df.columns)

print(str(correlationdf))

[[1.0, 0.2621831106319073, -0.3498870361913601, -0.22754243407728567, 0.0054380645902345455, -0.08019303979120941, 0.3830011912646053, 0.40008030107065345, -0.2074349414957676, -0.19449595631016842, -0.1365165732413513, 0.027584630997934137, 0.044356144422191414, 0.14511745368469148, 0.10922168988435187], [0.2621831106319073, 1.0, 0.13173030369957126, -0.14588636884853523, -0.06867695829634854, -0.013864625298016561, 0.05423039311298662, 0.20594058058911732, -0.047859093516036115, -0.3533721796161199, 0.1137716383392065, -0.08836282189613677, -0.07536879161747877, 0.4521957020132829, -0.05652709507509265], [-0.3498870361913601, 0.13173030369957126, 1.0, 0.6709789169180506, -0.0006684910327938819, -0.007840034749749248, -0.047128411649494224, -0.44030066714128646, 0.45597480449371997, -0.053134513277109365, 0.4997179503526042, 0.019029261406040162, 0.07242430866575386, 0.3531406085926749, -0.0002211602598921216], [-0.22754243407728567, -0.14588636884853523, 0.6709789169180506, 1.0, -0.03942517635646372, -0.039224314166830775, 0.05698412148423866, -0.2312259991265853, 0.41873045924203905, -0.06299658296246183, 0.47068801697888657, 0.06575970132037066, 0.12173839071563679, 0.21318262026110196, 0.10430385050049072], [0.0054380645902345455, -0.06867695829634854, -0.0006684910327938819, -0.03942517635646372, 1.0, -0.08944495602065969, 0.027820236082698905, 0.0071141976200455575, -0.0385215697460393, 0.10331479865408055, -0.09070928085101426, 0.024748899704094145, 0.03096832371809393, -0.03371261914210398, 0.029025751174332794], [-0.08019303979120941, -0.013864625298016561, -0.007840034749749248, -0.039224314166830775, -0.08944495602065969, 1.0, -0.014090599955156775, 0.007201276005109092, -0.044947424718673024, 0.09326692942934192, -0.08893945205648729, 0.026695064564031416, 0.05125389471822205, -0.10247990374834987, -0.14797083444555673], [0.3830011912646053, 0.05423039311298662, -0.047128411649494224, 0.05698412148423866, 0.027820236082698905, -0.014090599955156775, 1.0, 0.10891727933104071, -0.13955243704119408, -0.014208321999764613, -0.14415024891761588, 0.11375649558100549, 0.15608755559489612, 0.20637596757100318, 0.07209634709422777], [0.40008030107065345, 0.20594058058911732, -0.44030066714128646, -0.2312259991265853, 0.0071141976200455575, 0.007201276005109092, 0.10891727933104071, 1.0, -0.10377040985752758, -0.10314335007853181, -0.07610294592143618, -0.028276589052987768, 0.1053787574564542, -0.1227802162413711, 0.0188813529331215], [-0.2074349414957676, -0.047859093516036115, 0.45597480449371997, 0.41873045924203905, -0.0385215697460393, -0.044947424718673024, -0.13955243704119408, -0.10377040985752758, 1.0, 0.3068018347575882, 0.9041169005730315, 0.015789176536920267, -0.19851120983207446, 0.3786636066626807, 0.07222458038211689], [-0.19449595631016842, -0.3533721796161199, -0.053134513277109365, -0.06299658296246183, 0.10331479865408055, 0.09326692942934192, -0.014208321999764613, -0.10314335007853181, 0.3068018347575882, 1.0, -0.12104877689385991, 0.1444204756427707, 0.003271390756787971, 0.1699510782071378, -0.010498375133107486], [-0.1365165732413513, 0.1137716383392065, 0.4997179503526042, 0.47068801697888657, -0.09070928085101426, -0.08893945205648729, -0.14415024891761588, -0.07610294592143618, 0.9041169005730315, -0.12104877689385991, 1.0, -0.04971947911291968, -0.19600648326867134, 0.32397779378960584, 0.07942817977307376], [0.027584630997934137, -0.08836282189613677, 0.019029261406040162, 0.06575970132037066, 0.024748899704094145, 0.026695064564031416, 0.11375649558100549, -0.028276589052987768, 0.015789176536920267, 0.1444204756427707, -0.04971947911291968, 1.0, -0.008936423141141042, 0.06506734280385393, -0.12424798409789116], [0.044356144422191414, -0.07536879161747877, 0.07242430866575386, 0.12173839071563679, 0.03096832371809393, 0.05125389471822205, 0.15608755559489612, 0.1053787574564542, -0.19851120983207446, 0.003271390756787971, -0.19600648326867134, -0.008936423141141042, 1.0, -0.11167800453514784, 0.11674811883330147], [0.14511745368469148, 0.4521957020132829, 0.3531406085926749, 0.21318262026110196, -0.03371261914210398, -0.10247990374834987, 0.20637596757100318, -0.1227802162413711, 0.3786636066626807, 0.1699510782071378, 0.32397779378960584, 0.06506734280385393, -0.11167800453514784, 1.0, 0.05183802776388215], [0.10922168988435187, -0.05652709507509265, -0.0002211602598921216, 0.10430385050049072, 0.029025751174332794, -0.14797083444555673, 0.07209634709422777, 0.0188813529331215, 0.07222458038211689, -0.010498375133107486, 0.07942817977307376, -0.12424798409789116, 0.11674811883330147, 0.05183802776388215, 1.0]]

DataFrame[Transportation expense: double, Distance from Residence to Work: double, Service Time: double, Age: double, Work load Average/Day : double, Hit target: double, Children: double, Pet: double, Weight: double, Height: double, Body Mass Index: double, Absenteeism time in hours: double, Social Smoker: double, Social Drinker: double, Disciplinary failure: double]

In [44]:

df.stat.corr("Transportation expense", "Absenteeism time in hours")

Out[44]:

0.027584630997934092

In [3]:

# https://www.datasciencemadesimple.com/frequency-table-or-cross-table-in-pyspark-2-way-cross-table/

df.crosstab('Reason for Absence', 'Social drinker').show(28)

+---------------------------------+-----+----+

|Reason for Absence\_Social drinker|false|true|

+---------------------------------+-----+----+

| II| 1| 0|

| X| 11| 14|

| VII| 9| 6|

| XII| 4| 4|

| XV| 0| 2|

| XVIII| 12| 9|

| XXI| 4| 2|

| XXVII| 31| 38|

| XVII| 1| 0|

| XXVI| 8| 25|

| XI| 15| 11|

| IV| 1| 1|

| XVI| 3| 0|

| I| 11| 5|

| VIII| 5| 1|

| XIII| 21| 34|

| XIX| 11| 29|

| V| 1| 2|

| III| 0| 1|

| 0| 14| 29|

| XXVIII| 25| 87|

| XXV| 25| 6|

| XXII| 12| 26|

| VI| 5| 3|

| XXIII| 82| 67|

| XXIV| 0| 3|

| XIV| 6| 13|

| IX| 2| 2|

+---------------------------------+-----+----+

In [4]:

df.crosstab('Reason for Absence', 'Social smoker').show(28)

+--------------------------------+-----+----+

|Reason for Absence\_Social smoker|false|true|

+--------------------------------+-----+----+

| II| 1| 0|

| X| 24| 1|

| VII| 13| 2|

| XII| 8| 0|

| XV| 1| 1|

| XVIII| 17| 4|

| XXI| 4| 2|

| XXVII| 69| 0|

| XVII| 0| 1|

| XXVI| 33| 0|

| XI| 23| 3|

| IV| 2| 0|

| XVI| 1| 2|

| I| 15| 1|

| VIII| 4| 2|

| XIII| 54| 1|

| XIX| 36| 4|

| V| 3| 0|

| III| 1| 0|

| 0| 35| 8|

| XXVIII| 108| 4|

| XXV| 24| 7|

| XXII| 34| 4|

| VI| 8| 0|

| XXIII| 145| 4|

| XXIV| 3| 0|

| XIV| 16| 3|

| IX| 4| 0|

+--------------------------------+-----+----+

In [5]:

df.crosstab('Reason for Absence', 'Day of the week').show(28)

+----------------------------------+---+---+-----+----+---+

|Reason for Absence\_Day of the week|FRI|MON|THURS|TUES|WED|

+----------------------------------+---+---+-----+----+---+

| II| 0| 0| 0| 0| 1|

| X| 5| 9| 4| 1| 6|

| VII| 1| 7| 1| 1| 5|

| XII| 1| 2| 1| 2| 2|

| XV| 1| 1| 0| 0| 0|

| XVIII| 2| 5| 2| 8| 4|

| XXI| 0| 1| 2| 1| 2|

| XXVII| 20| 10| 12| 7| 20|

| XVII| 0| 0| 0| 1| 0|

| XXVI| 8| 10| 2| 6| 7|

| XI| 1| 10| 4| 4| 7|

| IV| 0| 0| 2| 0| 0|

| XVI| 1| 0| 0| 2| 0|

| I| 1| 6| 0| 3| 6|

| VIII| 1| 1| 1| 2| 1|

| XIII| 7| 12| 12| 10| 14|

| XIX| 7| 11| 8| 9| 5|

| V| 1| 0| 0| 1| 1|

| III| 0| 0| 0| 0| 1|

| 0| 6| 6| 7| 13| 11|

| XXVIII| 34| 22| 17| 26| 13|

| XXV| 5| 10| 8| 5| 3|

| XXII| 10| 9| 4| 6| 9|

| VI| 1| 2| 1| 2| 2|

| XXIII| 27| 22| 30| 37| 33|

| XXIV| 0| 1| 0| 1| 1|

| XIV| 4| 3| 7| 4| 1|

| IX| 0| 1| 0| 2| 1|

+----------------------------------+---+---+-----+----+---+

In [6]:

df.crosstab('Reason for Absence', 'Seasons').show(28)

+--------------------------+------+------+------+------+

|Reason for Absence\_Seasons|Autumn|Spring|Summer|Winter|

+--------------------------+------+------+------+------+

| II| 0| 1| 0| 0|

| X| 4| 7| 2| 12|

| VII| 4| 4| 2| 5|

| XII| 4| 2| 0| 2|

| XV| 0| 1| 1| 0|

| XVIII| 8| 4| 7| 2|

| XXI| 1| 3| 2| 0|

| XXVII| 56| 1| 0| 12|

| XVII| 1| 0| 0| 0|

| XXVI| 6| 11| 11| 5|

| XI| 4| 4| 9| 9|

| IV| 0| 2| 0| 0|

| XVI| 0| 0| 0| 3|

| I| 4| 6| 5| 1|

| VIII| 1| 1| 2| 2|

| XIII| 14| 11| 8| 22|

| XIX| 10| 4| 10| 16|

| V| 0| 0| 2| 1|

| III| 1| 0| 0| 0|

| 0| 2| 21| 6| 14|

| XXVIII| 18| 32| 28| 34|

| XXV| 14| 3| 11| 3|

| XXII| 9| 7| 11| 11|

| VI| 1| 3| 3| 1|

| XXIII| 27| 57| 43| 22|

| XXIV| 0| 2| 1| 0|

| XIV| 3| 7| 3| 6|

| IX| 0| 1| 3| 0|

+--------------------------+------+------+------+------+

In [2]:

# Encode the Categorical Variables

# String Indexer does not work for Boolean - need to treat separately

#

from pyspark.ml import Pipeline

from pyspark.ml.feature import StringIndexer

indexers = [StringIndexer(inputCol='Day of the week', outputCol='Day\_Encode'), StringIndexer(inputCol='Reason for absence', outputCol='Reason Encode'),

StringIndexer(inputCol='Seasons', outputCol='Season Encode'), StringIndexer(inputCol='Education', outputCol='Education Encode')]

pipeline = Pipeline(stages=indexers)

new\_df = pipeline.fit(df).transform(df)

new\_df.show()

+---+------------------+----------------+---------------+-------+----------------------+-------------------------------+------------+---+----------------------+----------+--------------------+-----------+--------+--------------+-------------+---+------+------+---------------+-------------------------+----------+-------------+-------------+----------------+

| ID|Reason for absence|Month of absence|Day of the week|Seasons|Transportation expense|Distance from Residence to Work|Service time|Age|Work load Average/day |Hit target|Disciplinary failure| Education|Children|Social drinker|Social smoker|Pet|Weight|Height|Body mass index|Absenteeism time in hours|Day\_Encode|Reason Encode|Season Encode|Education Encode|

+---+------------------+----------------+---------------+-------+----------------------+-------------------------------+------------+---+----------------------+----------+--------------------+-----------+--------+--------------+-------------+---+------+------+---------------+-------------------------+----------+-------------+-------------+----------------+

| 11| XXVI| JUL| TUES| Summer| 289| 36| 13| 33| 239.554| 97| false|High School| 2| true| false| 1| 90| 172| 30| 4| 2.0| 7.0| 3.0| 0.0|

| 36| 0| JUL| TUES| Summer| 118| 13| 18| 50| 239.554| 97| true|High School| 1| true| false| 0| 98| 178| 31| 0| 2.0| 4.0| 3.0| 0.0|

| 3| XXIII| JUL| WED| Summer| 179| 51| 18| 38| 239.554| 97| false|High School| 0| true| false| 0| 89| 170| 31| 2| 1.0| 0.0| 3.0| 0.0|

| 7| VII| JUL| THURS| Summer| 279| 5| 14| 39| 239.554| 97| false|High School| 2| true| true| 0| 68| 168| 24| 4| 4.0| 14.0| 3.0| 0.0|

| 11| XXIII| JUL| THURS| Summer| 289| 36| 13| 33| 239.554| 97| false|High School| 2| true| false| 1| 90| 172| 30| 2| 4.0| 0.0| 3.0| 0.0|

| 3| XXIII| JUL| FRI| Summer| 179| 51| 18| 38| 239.554| 97| false|High School| 0| true| false| 0| 89| 170| 31| 2| 3.0| 0.0| 3.0| 0.0|

| 10| XXII| JUL| FRI| Summer| 361| 52| 3| 28| 239.554| 97| false|High School| 1| true| false| 4| 80| 172| 27| 8| 3.0| 6.0| 3.0| 0.0|

| 20| XXIII| JUL| FRI| Summer| 260| 50| 11| 36| 239.554| 97| false|High School| 4| true| false| 0| 65| 168| 23| 4| 3.0| 0.0| 3.0| 0.0|

| 14| XIX| JUL| MON| Summer| 155| 12| 14| 34| 239.554| 97| false|High School| 2| true| false| 0| 95| 196| 25| 40| 0.0| 5.0| 3.0| 0.0|

| 1| XXII| JUL| MON| Summer| 235| 11| 14| 37| 239.554| 97| false| Postgrad| 1| false| false| 1| 88| 172| 29| 8| 0.0| 6.0| 3.0| 1.0|

| 20| I| JUL| MON| Summer| 260| 50| 11| 36| 239.554| 97| false|High School| 4| true| false| 0| 65| 168| 23| 8| 0.0| 13.0| 3.0| 0.0|

| 20| I| JUL| TUES| Summer| 260| 50| 11| 36| 239.554| 97| false|High School| 4| true| false| 0| 65| 168| 23| 8| 2.0| 13.0| 3.0| 0.0|

| 20| XI| JUL| WED| Summer| 260| 50| 11| 36| 239.554| 97| false|High School| 4| true| false| 0| 65| 168| 23| 8| 1.0| 9.0| 3.0| 0.0|

| 3| XI| JUL| WED| Summer| 179| 51| 18| 38| 239.554| 97| false|High School| 0| true| false| 0| 89| 170| 31| 1| 1.0| 9.0| 3.0| 0.0|

| 3| XXIII| JUL| WED| Summer| 179| 51| 18| 38| 239.554| 97| false|High School| 0| true| false| 0| 89| 170| 31| 4| 1.0| 0.0| 3.0| 0.0|

| 24| XIV| JUL| FRI| Summer| 246| 25| 16| 41| 239.554| 97| false|High School| 0| true| false| 0| 67| 170| 23| 8| 3.0| 12.0| 3.0| 0.0|

| 3| XXIII| JUL| FRI| Summer| 179| 51| 18| 38| 239.554| 97| false|High School| 0| true| false| 0| 89| 170| 31| 2| 3.0| 0.0| 3.0| 0.0|

| 3| XXI| JUL| MON| Summer| 179| 51| 18| 38| 239.554| 97| false|High School| 0| true| false| 0| 89| 170| 31| 8| 0.0| 18.0| 3.0| 0.0|

| 6| XI| JUL| THURS| Summer| 189| 29| 13| 33| 239.554| 97| false|High School| 2| false| false| 2| 69| 167| 25| 8| 4.0| 9.0| 3.0| 0.0|

| 33| XXIII| AUG| WED| Summer| 248| 25| 14| 47| 205.917| 92| false|High School| 2| false| false| 1| 86| 165| 32| 2| 1.0| 0.0| 3.0| 0.0|

+---+------------------+----------------+---------------+-------+----------------------+-------------------------------+------------+---+----------------------+----------+--------------------+-----------+--------+--------------+-------------+---+------+------+---------------+-------------------------+----------+-------------+-------------+----------------+

only showing top 20 rows

In [4]:

# https://towardsdatascience.com/your-first-apache-spark-ml-model-d2bb82b599dd

# Drop categorical variables

new\_df = new\_df.drop('Day of the week')

new\_df = new\_df.drop('Reason for absence')

new\_df = new\_df.drop('Seasons')

new\_df = new\_df.drop('Education')

new\_df = new\_df.drop('Month of absence')

new\_df.show()

+---+----------------------+-------------------------------+------------+---+----------------------+----------+--------------------+--------+--------------+-------------+---+------+------+---------------+-------------------------+----------+-------------+-------------+----------------+

| ID|Transportation expense|Distance from Residence to Work|Service time|Age|Work load Average/day |Hit target|Disciplinary failure|Children|Social drinker|Social smoker|Pet|Weight|Height|Body mass index|Absenteeism time in hours|Day\_Encode|Reason Encode|Season Encode|Education Encode|

+---+----------------------+-------------------------------+------------+---+----------------------+----------+--------------------+--------+--------------+-------------+---+------+------+---------------+-------------------------+----------+-------------+-------------+----------------+

| 11| 289| 36| 13| 33| 239.554| 97| false| 2| true| false| 1| 90| 172| 30| 4| 2.0| 7.0| 3.0| 0.0|

| 36| 118| 13| 18| 50| 239.554| 97| true| 1| true| false| 0| 98| 178| 31| 0| 2.0| 4.0| 3.0| 0.0|

| 3| 179| 51| 18| 38| 239.554| 97| false| 0| true| false| 0| 89| 170| 31| 2| 1.0| 0.0| 3.0| 0.0|

| 7| 279| 5| 14| 39| 239.554| 97| false| 2| true| true| 0| 68| 168| 24| 4| 4.0| 14.0| 3.0| 0.0|

| 11| 289| 36| 13| 33| 239.554| 97| false| 2| true| false| 1| 90| 172| 30| 2| 4.0| 0.0| 3.0| 0.0|

| 3| 179| 51| 18| 38| 239.554| 97| false| 0| true| false| 0| 89| 170| 31| 2| 3.0| 0.0| 3.0| 0.0|

| 10| 361| 52| 3| 28| 239.554| 97| false| 1| true| false| 4| 80| 172| 27| 8| 3.0| 6.0| 3.0| 0.0|

| 20| 260| 50| 11| 36| 239.554| 97| false| 4| true| false| 0| 65| 168| 23| 4| 3.0| 0.0| 3.0| 0.0|

| 14| 155| 12| 14| 34| 239.554| 97| false| 2| true| false| 0| 95| 196| 25| 40| 0.0| 5.0| 3.0| 0.0|

| 1| 235| 11| 14| 37| 239.554| 97| false| 1| false| false| 1| 88| 172| 29| 8| 0.0| 6.0| 3.0| 1.0|

| 20| 260| 50| 11| 36| 239.554| 97| false| 4| true| false| 0| 65| 168| 23| 8| 0.0| 13.0| 3.0| 0.0|

| 20| 260| 50| 11| 36| 239.554| 97| false| 4| true| false| 0| 65| 168| 23| 8| 2.0| 13.0| 3.0| 0.0|

| 20| 260| 50| 11| 36| 239.554| 97| false| 4| true| false| 0| 65| 168| 23| 8| 1.0| 9.0| 3.0| 0.0|

| 3| 179| 51| 18| 38| 239.554| 97| false| 0| true| false| 0| 89| 170| 31| 1| 1.0| 9.0| 3.0| 0.0|

| 3| 179| 51| 18| 38| 239.554| 97| false| 0| true| false| 0| 89| 170| 31| 4| 1.0| 0.0| 3.0| 0.0|

| 24| 246| 25| 16| 41| 239.554| 97| false| 0| true| false| 0| 67| 170| 23| 8| 3.0| 12.0| 3.0| 0.0|

| 3| 179| 51| 18| 38| 239.554| 97| false| 0| true| false| 0| 89| 170| 31| 2| 3.0| 0.0| 3.0| 0.0|

| 3| 179| 51| 18| 38| 239.554| 97| false| 0| true| false| 0| 89| 170| 31| 8| 0.0| 18.0| 3.0| 0.0|

| 6| 189| 29| 13| 33| 239.554| 97| false| 2| false| false| 2| 69| 167| 25| 8| 4.0| 9.0| 3.0| 0.0|

| 33| 248| 25| 14| 47| 205.917| 92| false| 2| false| false| 1| 86| 165| 32| 2| 1.0| 0.0| 3.0| 0.0|

+---+----------------------+-------------------------------+------------+---+----------------------+----------+--------------------+--------+--------------+-------------+---+------+------+---------------+-------------------------+----------+-------------+-------------+----------------+

only showing top 20 rows

In [5]:

# Assemble features into vector

from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols=new\_df.columns, outputCol="features", handleInvalid='keep')

features\_df = assembler.transform(new\_df).select("features")

features\_df.show()

+--------------------+

| features|

+--------------------+

|[11.0,289.0,36.0,...|

|[36.0,118.0,13.0,...|

|[3.0,179.0,51.0,1...|

|[7.0,279.0,5.0,14...|

|[11.0,289.0,36.0,...|

|[3.0,179.0,51.0,1...|

|[10.0,361.0,52.0,...|

|[20.0,260.0,50.0,...|

|[14.0,155.0,12.0,...|

|[1.0,235.0,11.0,1...|

|[20.0,260.0,50.0,...|

|[20.0,260.0,50.0,...|

|[20.0,260.0,50.0,...|

|[3.0,179.0,51.0,1...|

|[3.0,179.0,51.0,1...|

|[24.0,246.0,25.0,...|

|[3.0,179.0,51.0,1...|

|[3.0,179.0,51.0,1...|

|[6.0,189.0,29.0,1...|

|[33.0,248.0,25.0,...|

+--------------------+

only showing top 20 rows

In [31]:

# Scale using Robust Scaler - noting high standard deviations with majority of numeric variables

# https://spark.apache.org/docs/latest/ml-features#robustscaler

from pyspark.ml.feature import RobustScaler

scaler = RobustScaler(inputCol='features', outputCol='scaledFeatures')

data\_scale=scaler.fit(features\_df)

scaled\_features=data\_scale.transform(features\_df)

scaled\_features.show() # Scaled values

+--------------------+--------------------+

| features| scaledFeatures|

+--------------------+--------------------+

|[11.0,289.0,36.0,...|[0.57894736842105...|

|[36.0,118.0,13.0,...|[1.89473684210526...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|

|[7.0,279.0,5.0,14...|[0.36842105263157...|

|[11.0,289.0,36.0,...|[0.57894736842105...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|

|[10.0,361.0,52.0,...|[0.52631578947368...|

|[20.0,260.0,50.0,...|[1.05263157894736...|

|[14.0,155.0,12.0,...|[0.73684210526315...|

|[1.0,235.0,11.0,1...|[0.05263157894736...|

|[20.0,260.0,50.0,...|[1.05263157894736...|

|[20.0,260.0,50.0,...|[1.05263157894736...|

|[20.0,260.0,50.0,...|[1.05263157894736...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|

|[24.0,246.0,25.0,...|[1.26315789473684...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|

|[6.0,189.0,29.0,1...|[0.31578947368421...|

|[33.0,248.0,25.0,...|[1.73684210526315...|

+--------------------+--------------------+

only showing top 20 rows

In [65]:

# PCA for dimensionality reduction

# https://spark.apache.org/docs/latest/ml-features.html#pca

from pyspark.ml.feature import PCA

pca = PCA(k=10, inputCol='scaledFeatures', outputCol='PCAFeatures')

model = pca.fit(scaled\_features)

result = model.transform(scaled\_features)

result.show()

+--------------------+--------------------+--------------------+

| features| scaledFeatures| PCAFeatures|

+--------------------+--------------------+--------------------+

|[11.0,289.0,36.0,...|[0.57894736842105...|[-29.518529288652...|

|[36.0,118.0,13.0,...|[1.89473684210526...|[-30.089543168529...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|[-28.913580202213...|

|[7.0,279.0,5.0,14...|[0.36842105263157...|[-28.997945944158...|

|[11.0,289.0,36.0,...|[0.57894736842105...|[-29.109828840173...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|[-28.873109235850...|

|[10.0,361.0,52.0,...|[0.52631578947368...|[-29.783740158974...|

|[20.0,260.0,50.0,...|[1.05263157894736...|[-28.814946737301...|

|[14.0,155.0,12.0,...|[0.73684210526315...|[-38.777914487387...|

|[1.0,235.0,11.0,1...|[0.05263157894736...|[-30.140669118281...|

|[20.0,260.0,50.0,...|[1.05263157894736...|[-29.600487321591...|

|[20.0,260.0,50.0,...|[1.05263157894736...|[-29.560016355228...|

|[20.0,260.0,50.0,...|[1.05263157894736...|[-29.533752520459...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|[-28.874775829749...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|[-29.200435877917...|

|[24.0,246.0,25.0,...|[1.26315789473684...|[-29.869866243987...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|[-28.873109235850...|

|[3.0,179.0,51.0,1...|[0.15789473684210...|[-30.003629643281...|

|[6.0,189.0,29.0,1...|[0.31578947368421...|[-29.186643234199...|

|[33.0,248.0,25.0,...|[1.73684210526315...|[-27.987609846477...|

+--------------------+--------------------+--------------------+

only showing top 20 rows

In [36]:

# https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.PCA.html

# PCA 1 = 31%, PCA 2 = 22%, PCA 3 = 12%, PCA 4 = 6%, PCA 5 = 6% .... 94% covered in 10 components

model.explainedVariance

Out[36]:

DenseVector([0.309, 0.2216, 0.1201, 0.0633, 0.0579, 0.0463, 0.0378, 0.0336, 0.0283, 0.0222])

In [66]:

# https://towardsdatascience.com/k-means-clustering-using-pyspark-on-big-data-6214beacdc8b

from pyspark.ml.clustering import KMeans

from pyspark.ml.evaluation import ClusteringEvaluator

silhouette\_score=[]

evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='PCAFeatures', metricName='silhouette')

for i in range(2,10):

KMeans\_model=KMeans(featuresCol='PCAFeatures', k=i)

KMeans\_fit=KMeans\_model.fit(result)

output=KMeans\_fit.transform(result)

score=evaluator.evaluate(output)

silhouette\_score.append(score)

print("Silhouette Score:",score)

Silhouette Score: 0.7234172790092293

Silhouette Score: 0.33160803606310213

Silhouette Score: 0.24405840925051367

Silhouette Score: 0.25953821077188277

Silhouette Score: 0.2494680505931106

Silhouette Score: 0.24741753680948236

Silhouette Score: 0.24291180470239282

Silhouette Score: 0.28328919571540345

In [67]:

# Visualise scores

import matplotlib.pyplot as plt

fig, axis=plt.subplots(1,1, figsize=(8,6))

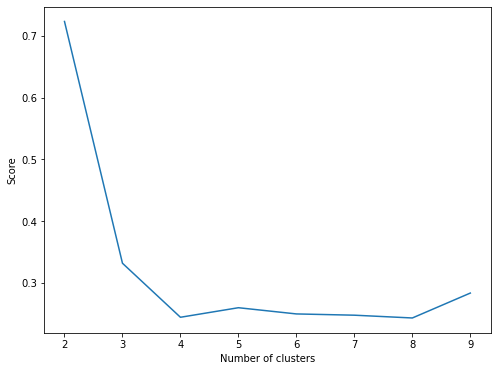
axis.plot(range(2,10), silhouette\_score)

axis.set\_xlabel('Number of clusters')

axis.set\_ylabel('Score')

Out[67]:

Text(0, 0.5, 'Score')



In [70]:

# Suggesting 9 clusters?

silhouette\_score=[]

evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='PCAFeatures', metricName='silhouette')

for i in range(2,15):

KMeans\_model=KMeans(featuresCol='PCAFeatures', k=i)

KMeans\_fit=KMeans\_model.fit(result)

output=KMeans\_fit.transform(result)

score=evaluator.evaluate(output)

silhouette\_score.append(score)

print("Silhouette Score:",score)

fig, axis=plt.subplots(1,1, figsize=(8,6))

axis.plot(range(2,15), silhouette\_score)

axis.set\_xlabel('Number of clusters')

axis.set\_ylabel('Score')

Silhouette Score: 0.7234172790092293

Silhouette Score: 0.33160803606310213

Silhouette Score: 0.24405840925051367

Silhouette Score: 0.25953821077188277

Silhouette Score: 0.2494680505931106

Silhouette Score: 0.24741753680948236

Silhouette Score: 0.24291180470239282

Silhouette Score: 0.28328919571540345

Silhouette Score: 0.3105733074347437

Silhouette Score: 0.31949857857372493

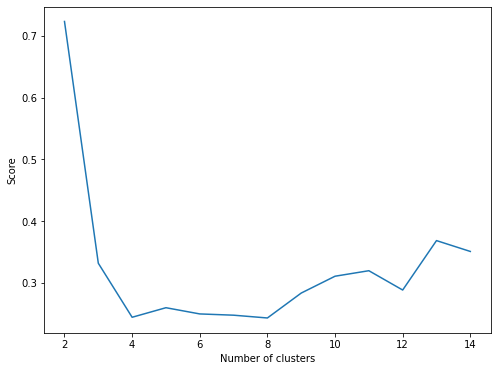
Silhouette Score: 0.28819886777368264

Silhouette Score: 0.3682986803166066

Silhouette Score: 0.35071589143070475

Out[70]:

Text(0, 0.5, 'Score')



In [71]:

# Without PCA

silhouette\_score=[]

evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='scaledFeatures', metricName='silhouette')

for i in range(2,10):

KMeans\_model=KMeans(featuresCol='scaledFeatures', k=i)

KMeans\_fit=KMeans\_model.fit(scaled\_features)

output=KMeans\_fit.transform(scaled\_features)

score=evaluator.evaluate(output)

silhouette\_score.append(score)

print("Silhouette Score:",score)

Silhouette Score: 0.7091755657691248

Silhouette Score: 0.1784007996447161

Silhouette Score: 0.33900099388581606

Silhouette Score: 0.23185693917442285

Silhouette Score: 0.21866165252280284

Silhouette Score: 0.20845857182353553

Silhouette Score: 0.21222832763647184

Silhouette Score: 0.21309655000705022

In [72]:

fig, axis=plt.subplots(1,1, figsize=(8,6))

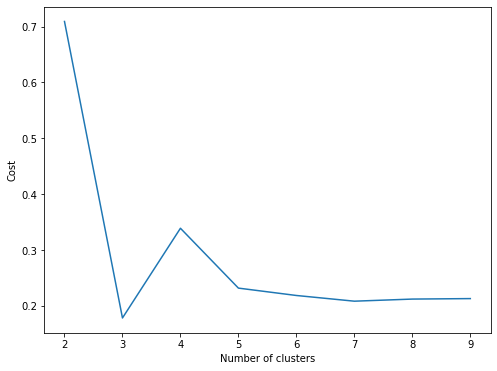
axis.plot(range(2,10), silhouette\_score)

axis.set\_xlabel('Number of clusters')

axis.set\_ylabel('Cost')

Out[72]:

Text(0, 0.5, 'Cost')



In [ ]: