```
from google.colab import files
upload = files.upload()
```

Choose Files RiceSeed.csv

- **RiceSeed.csv**(text/csv) - 1991236 bytes, last modified: 2/5/2025 - 100% done

```
Saving RiceSeed.csv to RiceSeed (1).csv
```

```
import pandas as pd
df = pd.read_csv('RiceSeed.csv')
df
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
0	1	4537	92.229316	64.012769	0.719916	4677	76.004525	0.657536	273.085	0.764510	1
1	2	2872	74.691881	51.400454	0.725553	3015	60.471018	0.713009	208.317	0.831658	1
2	3	3048	76.293164	52.043491	0.731211	3132	62.296341	0.759153	210.012	0.868434	1
3	4	3073	77.033628	51.928487	0.738639	3157	62.551300	0.783529	210.657	0.870203	1
4	5	3693	85.124785	56.374021	0.749282	3802	68.571668	0.769375	230.332	0.874743	1
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

18185 rows × 12 columns

```
df.value_counts()
```

id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	AspectRation
1	**4537**	**92.229316**	**64.012769**	**0.719916**	**4677**	**76.004525**	**0.657536**	**273.085**	**0.764510**	**1.440796**
12122	**8457**	**153.057290**	**71.702046**	**0.883482**	**8738**	**103.767947**	**0.629616**	**373.417**	**0.762146**	**2.134629**
12128	**6387**	**133.223052**	**62.407478**	**0.883493**	**6656**	**90.178606**	**0.574474**	**321.425**	**0.776868**	**2.134729**
12127	**6597**	**134.797153**	**63.145261**	**0.883492**	**6745**	**91.649120**	**0.598639**	**324.897**	**0.785353**	**2.134715**
12126	**8060**	**148.873706**	**69.739469**	**0.883491**	**8265**	**101.303064**	**0.707763**	**358.818**	**0.786677**	**2.134712**
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
6069	**4709**	**135.187497**	**45.187593**	**0.942481**	**4832**	**77.431809**	**0.641903**	**302.556**	**0.646438**	**2.991695**
6070	**6033**	**153.523699**	**51.316550**	**0.942482**	**6235**	**87.643906**	**0.489374**	**347.177**	**0.628986**	**2.991700**
6071	**6613**	**160.105377**	**53.515361**	**0.942484**	**6784**	**91.760193**	**0.558059**	**354.890**	**0.659813**	**2.991765**
6072	**5647**	**147.940855**	**49.448724**	**0.942486**	**5763**	**84.793772**	**0.499293**	**326.435**	**0.665939**	**2.991803**
18185	**6134**	**153.081981**	**51.590606**	**0.941500**	**6283**	**88.374495**	**0.489975**	**338.613**	**0.672274**	**2.967245**

18185 rows × 1 columns

dtype: int64

```
print(df.columns)
```

```
Index(['id', 'Area', 'MajorAxisLength', 'MinorAxisLength', 'Eccentricity',
       'ConvexArea', 'EquivDiameter', 'Extent', 'Perimeter', 'Roundness',
       'AspectRation', 'Class'],
      dtype='object')
```

```
print(df.shape)
```

```
(18185, 12)
```

```
import numpy as np
def dataPreperation(df):

    df = df.select_dtypes(include=[np.number]).fillna(df.mean())
```

```
    for col in df.columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df2 = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
        if(len(df2['Class'].unique()) == 1):
            df2 = df
        else:
            df = df2
    if (df2.duplicated().sum() > 0):
        df2 = df2.drop_duplicates()

    return df2
```

```
df2 = dataPreperation(df)
df2
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
158	159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2
186	187	6946	136.303603	65.738475	0.876009	7115	94.042128	0.594437	338.782	0.760508	2
207	208	5786	125.555431	59.626250	0.880040	5945	85.831020	0.623491	307.304	0.769931	2
218	219	5896	127.013995	59.843292	0.882050	6092	86.643063	0.610352	309.264	0.774655	2
227	228	6082	129.098407	60.416315	0.883736	6204	87.999107	0.661375	308.106	0.805111	2
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

17597 rows × 12 columns

Next steps: Generate code with df2 | View recommended plots | New interactive sheet

```
df2.value_counts()
```

id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	AspectRation
159	**6374**	**129.576759**	**63.762307**	**0.870549**	**6476**	**90.086785**	**0.662234**	**314.656**	**0.809003**	**2.032184**
12305	**8534**	**153.288378**	**71.562448**	**0.884337**	**8784**	**104.239274**	**0.632851**	**369.860**	**0.783949**	**2.142023**
12311	**8644**	**154.426134**	**72.088637**	**0.884354**	**8911**	**104.908925**	**0.715563**	**375.375**	**0.770893**	**2.142170**
12310	**7826**	**147.598959**	**68.902461**	**0.884351**	**8052**	**99.821704**	**0.582595**	**355.794**	**0.776877**	**2.142144**
12309	**9738**	**163.629247**	**76.386364**	**0.884349**	**9916**	**111.349929**	**0.707703**	**395.174**	**0.783615**	**2.142126**
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
6405	**5548**	**147.101390**	**48.865420**	**0.943213**	**5710**	**84.047207**	**0.825472**	**331.484**	**0.634485**	**3.010337**
6406	**6145**	**155.564253**	**51.676350**	**0.943214**	**6297**	**88.453700**	**0.462587**	**344.030**	**0.652438**	**3.010357**
6407	**6104**	**153.961733**	**51.143402**	**0.943215**	**6244**	**88.158120**	**0.495696**	**341.131**	**0.659147**	**3.010393**
6408	**5822**	**151.880538**	**50.451950**	**0.943215**	**6038**	**86.097623**	**0.723589**	**340.143**	**0.632352**	**3.010400**
18185	**6134**	**153.081981**	**51.590606**	**0.941500**	**6283**	**88.374495**	**0.489975**	**338.613**	**0.672274**	**2.967245**

17597 rows × 1 columns

dtype: int64

```
df3 = dataPreperation(df2)
df3
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
158	159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2
186	187	6946	136.303603	65.738475	0.876009	7115	94.042128	0.594437	338.782	0.760508	2
218	219	5896	127.013995	59.843292	0.882050	6092	86.643063	0.610352	309.264	0.774655	2
227	228	6082	129.098407	60.416315	0.883736	6204	87.999107	0.661375	308.106	0.805111	2
242	243	6063	131.291578	60.178119	0.888769	6203	87.861547	0.789453	313.372	0.775849	2
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

17512 rows × 12 columns

Next steps: Generate code with df3 View recommended plots New interactive sheet

```
df4 = dataPreperation(df3)
df4
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
158	159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2
186	187	6946	136.303603	65.738475	0.876009	7115	94.042128	0.594437	338.782	0.760508	2
218	219	5896	127.013995	59.843292	0.882050	6092	86.643063	0.610352	309.264	0.774655	2
227	228	6082	129.098407	60.416315	0.883736	6204	87.999107	0.661375	308.106	0.805111	2
242	243	6063	131.291578	60.178119	0.888769	6203	87.861547	0.789453	313.372	0.775849	2
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

17492 rows × 12 columns

Next steps: Generate code with df4 View recommended plots New interactive sheet

```
df5 = dataPreperation(df4)
df5
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
158	159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2
186	187	6946	136.303603	65.738475	0.876009	7115	94.042128	0.594437	338.782	0.760508	2
218	219	5896	127.013995	59.843292	0.882050	6092	86.643063	0.610352	309.264	0.774655	2
227	228	6082	129.098407	60.416315	0.883736	6204	87.999107	0.661375	308.106	0.805111	2
242	243	6063	131.291578	60.178119	0.888769	6203	87.861547	0.789453	313.372	0.775849	2
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

17490 rows × 12 columns

Next steps: Generate code with df5 View recommended plots New interactive sheet

```
df6 = dataPreperation(df5)
df6
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
158	159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2
186	187	6946	136.303603	65.738475	0.876009	7115	94.042128	0.594437	338.782	0.760508	2
218	219	5896	127.013995	59.843292	0.882050	6092	86.643063	0.610352	309.264	0.774655	2
227	228	6082	129.098407	60.416315	0.883736	6204	87.999107	0.661375	308.106	0.805111	2
242	243	6063	131.291578	60.178119	0.888769	6203	87.861547	0.789453	313.372	0.775849	2
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

17490 rows × 12 columns

Next steps: Generate code with df6 | View recommended plots | New interactive sheet

```
X = df6.drop('Class', axis=1)
y = df6['Class']
```

```
y.value_counts()
```

Class	count
1	9463
0	8027

dtype: int64

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.3)
```

```
def runModel(model, X_train, X_test, y_train, y_test):
  model.fit(X_train, y_train)

  y_pred = model.predict(X_test)
  y_train_pred = model.predict(X_train)

  acc_test = accuracy_score(y_test, y_pred)
  acc_train = accuracy_score(y_train, y_train_pred)

  y_pred_proba = model.predict_proba(X_test)

  cm = confusion_matrix(y_test, y_pred)
  # cr = classification_report(y_test, y_pred)
  print(f'Train Accuracy: {acc_train}')
  print(f'Test Accuracy: {acc_test}')
  print(cm)
  return acc_test, acc_train, cm, y_pred_proba, y_pred
```

```
from sklearn.linear_model import LogisticRegression as LR
from sklearn.tree import DecisionTreeClassifier as DT
from xgboost import XGBClassifier as XGB
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
models_arr = [LR(max_iter=10000), DT(), XGB(), KNN(), RF()]
for model in models_arr:
  print(model.__class__.__name__)
  runModel(model, X_train, X_test, y_train, y_test)
```

```
LogisticRegression
Train Accuracy: 1.0
```

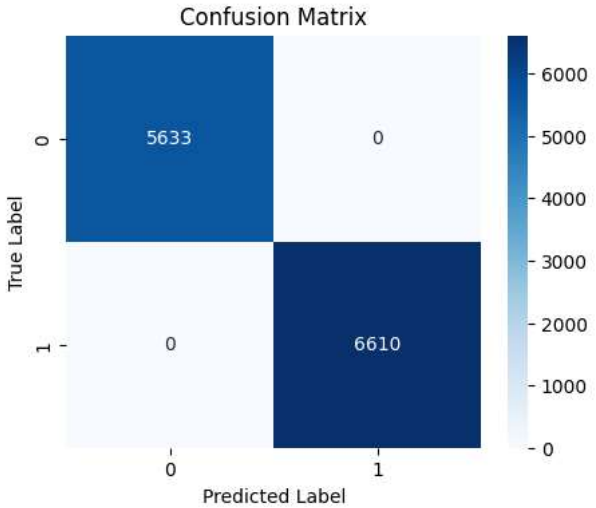
```
Test Accuracy: 1.0
[[5633    0]
 [   0 6610]]
DecisionTreeClassifier
Train Accuracy: 1.0
Test Accuracy: 1.0
[[5633    0]
 [   0 6610]]
XGBClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9994282447112636
[[5633    0]
 [   7 6603]]
KNeighborsClassifier
Train Accuracy: 0.9853249475890985
Test Accuracy: 0.9835007759536062
[[5470  163]
 [  39 6571]]
RandomForestClassifier
Train Accuracy: 1.0
Test Accuracy: 1.0
[[5633    0]
 [   0 6610]]
```

Generate visualize cm Close

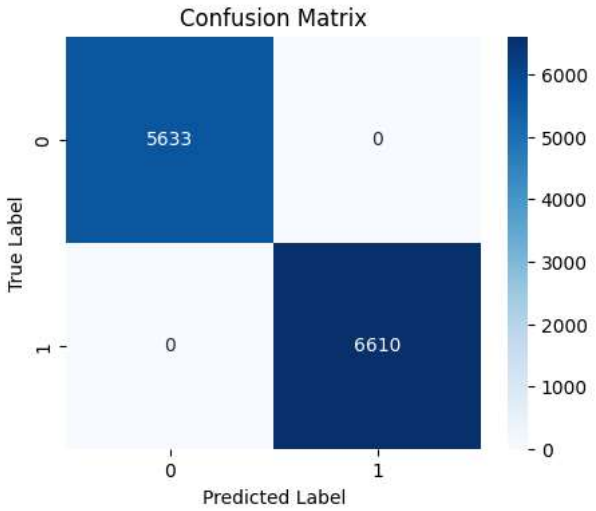
1 of 1 Use code with caution

```
# prompt: visualize cm

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
def runModel(model, X_train, X_test, y_train, y_test):
  model.fit(X_train, y_train)

  y_pred = model.predict(X_test)
  y_train_pred = model.predict(X_train)

  acc_test = accuracy_score(y_test, y_pred)
  acc_train = accuracy_score(y_train, y_train_pred)

  y_pred_proba = model.predict_proba(X_test)

  cm = confusion_matrix(y_test, y_pred)
  # cr = classification_report(y_test, y_pred)
  print(f'Train Accuracy: {acc_train}')
  print(f'Test Accuracy: {acc_test}')
  print(cm)
  plt.figure(figsize=(5,4))
  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
  plt.xlabel('Predicted Label')
  plt.ylabel('True Label')
  plt.title('Confusion Matrix')
  plt.show()
  return acc_test, acc_train, cm, y_pred_proba, y_pred


models_arr = [LR(max_iter=10000), DT(), XGB(), KNN(), RF()]
for model in models_arr:
  print(model.__class__.__name__)
  runModel(model, X_train, X_test, y_train, y_test)
```

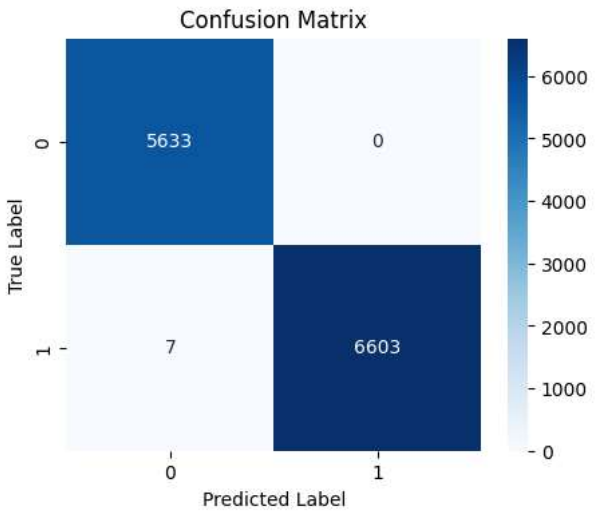
```
LogisticRegression
Train Accuracy: 1.0
Test Accuracy: 1.0
[[5633    0]
 [   0 6610]]
```



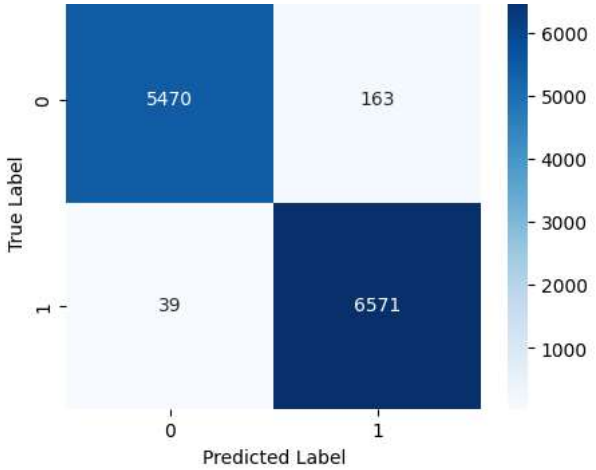

```
DecisionTreeClassifier
Train Accuracy: 1.0
Test Accuracy: 1.0
[[5633    0]
 [   0 6610]]
```




```
XGBClassifier
Train Accuracy: 1.0
Test Accuracy: 0.9994282447112636
[[5633    0]
 [   7 6603]]
```




```
KNeighborsClassifier
Train Accuracy: 0.9853249475890985
Test Accuracy: 0.9835007759536062
[[5470  163]
 [  39 6571]]
```




```
RandomForestClassifier
Train Accuracy: 1.0
Test Accuracy: 1.0
[[5633    0]
 [   0 6610]]
```


```
df6.value_counts()
```

id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	AspectRation
159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2.032184
12338	7981	148.500352	69.289564	0.884471	8151	100.805381	0.810336	355.940	0.791614	2.143185
12344	8282	151.786190	70.817611	0.884489	8452	102.688704	0.582911	367.648	0.769982	2.143340
12343	7419	143.597696	66.997704	0.884487	7654	97.191379	0.576815	349.344	0.763921	2.143323
12342	6893	138.154508	64.459970	0.884480	7018	93.682657	0.669483	333.511	0.778750	2.143261
...	...	...	...	...	...	...	...	...	...	...
6457	4661	134.666231	44.691413	0.943326	4772	77.036157	0.653900	298.040	0.659386	3.013246
6458	6018	153.705881	51.007240	0.943332	6195	87.534882	0.560857	342.476	0.644765	3.013413
6459	5207	143.368782	47.574139	0.943339	5357	81.423328	0.466786	321.814	0.631812	3.013586
6460	6457	159.372865	52.884165	0.943340	6616	90.671427	0.697074	356.700	0.637726	3.013622
18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2.967245

17490 rows × 1 columns

dtype: int64

```
df7 = dataPreperation(df6)
df7
```

	id	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	EquivDiameter	Extent	Perimeter	Roundness	Aspect
158	159	6374	129.576759	63.762307	0.870549	6476	90.086785	0.662234	314.656	0.809003	2
186	187	6946	136.303603	65.738475	0.876009	7115	94.042128	0.594437	338.782	0.760508	2
218	219	5896	127.013995	59.843292	0.882050	6092	86.643063	0.610352	309.264	0.774655	2
227	228	6082	129.098407	60.416315	0.883736	6204	87.999107	0.661375	308.106	0.805111	2
242	243	6063	131.291578	60.178119	0.888769	6203	87.861547	0.789453	313.372	0.775849	2
...	...	...	...	...	...	...	...	...	...	...	
18180	18181	5853	148.624571	51.029281	0.939210	6008	86.326537	0.498594	332.960	0.663444	2
18181	18182	7585	169.593996	58.141659	0.939398	7806	98.272692	0.647461	385.506	0.641362	2
18182	18183	6365	154.777085	52.908085	0.939760	6531	90.023162	0.561287	342.253	0.682832	2
18183	18184	5960	151.397924	51.474600	0.940427	6189	87.112041	0.492399	343.371	0.635227	2
18184	18185	6134	153.081981	51.590606	0.941500	6283	88.374495	0.489975	338.613	0.672274	2

17490 rows × 12 columns

Next steps: Generate code with df7 | View recommended plots | New interactive sheet