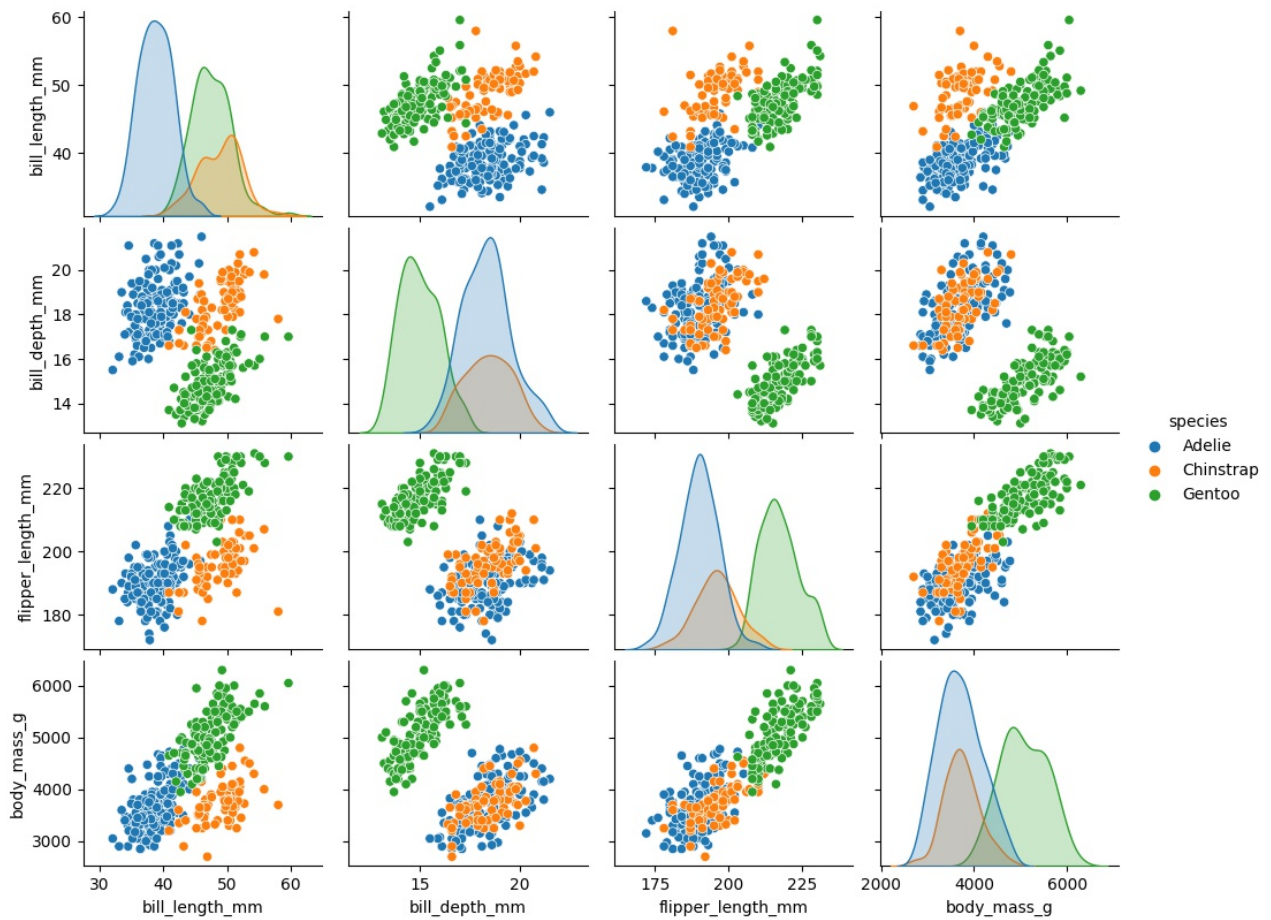


Version4
Status!
Author MK

Data Science Story 2

maXbox Starter 100_1



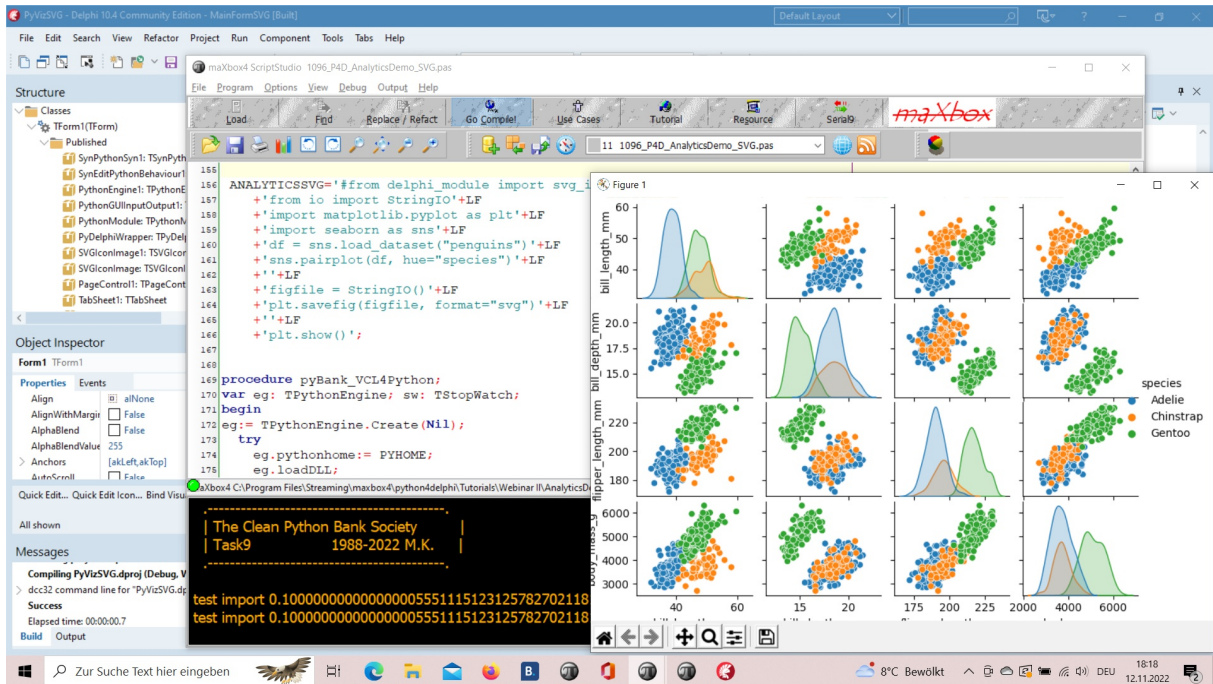
Links and Sources

Title

<https://www.kaggle.com/code/parulpandey/penguin-dataset-the-new-iris>

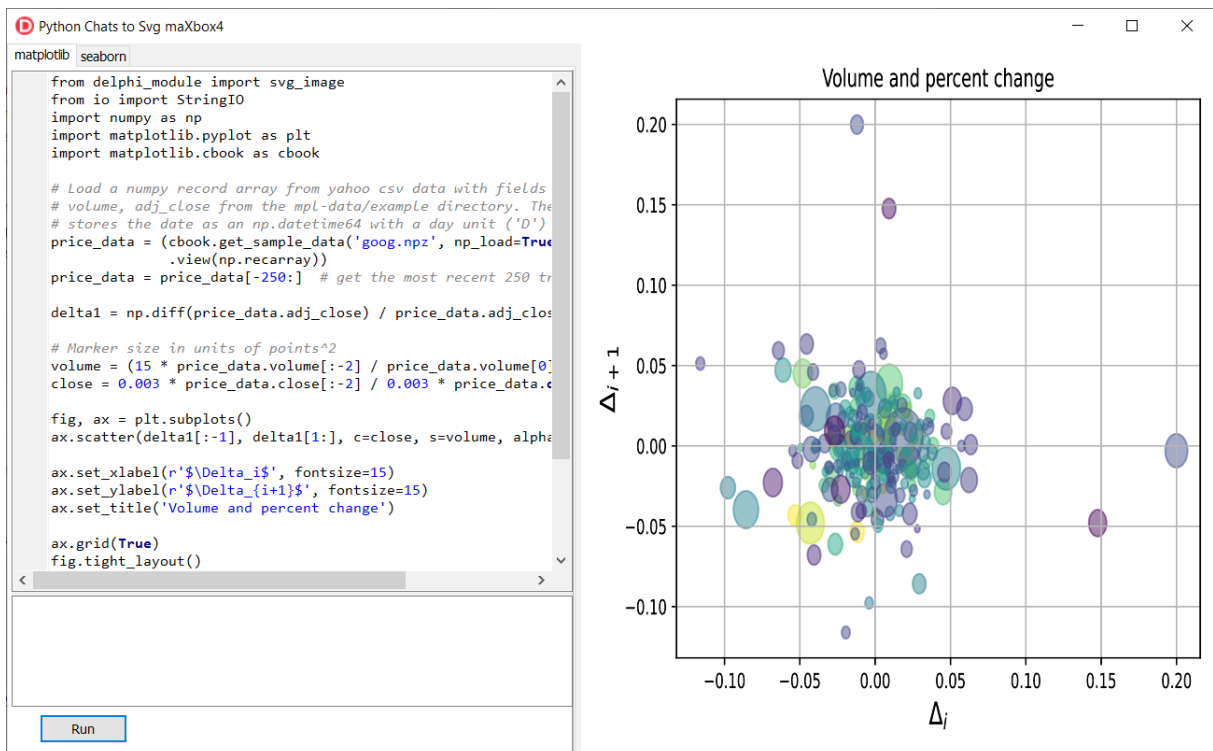
[python4delphi/Tutorials/Webinar II at master · maxkleiner/python4delphi \(github.com\)](python4delphi/Tutorials/Webinar II at master · maxkleiner/python4delphi (github.com))

<https://maxbox4.wordpress.com/>



1096_2022-11-12_svg_delphi_maxbox_seaborn4.png

RAD Studio 11.4, maXbox4 and Python 3.8 with Seaborn



Python4Delphi with SVG Plot

1 From Data to Plot

1.1 Functions

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper.

Palmer Archipelago (Antarctica) penguin dataset appears to be a drop in replacement for the same. It is a great intro dataset for data exploration & visualization. But the penguins dataset has different number of samples for each species. It can be observed that unlike the Iris dataset, this data contains different number of entries for each species.

What is culmen?

The upper margin of the beak or bill is referred to as the culmen and the measurement is taken using calipers with one jaw at the tip of the upper mandible and the other at base of the skull or the first feathers depending on the standard chosen.

First we get the data:

```
import seaborn as sns
df = sns.load_dataset("penguins")
```

Or alternate

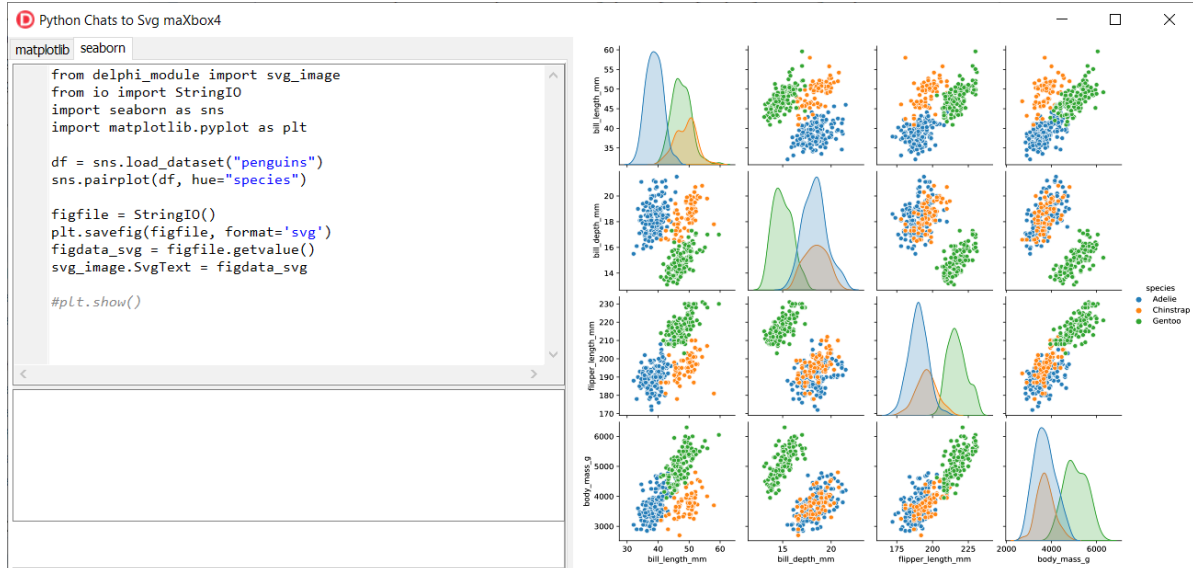
```
df = pd.read_csv('../input/palmer-archipelago-antarctica-penguin-
data/penguins_size.csv')
```

```
df.head()
```

	species	island	culmen_length_ mm	culmen_depth_ mm	flipper_length_ mm	body_mass_ g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

	species	island	culmen_length_ mm	culmen_depth_ mm	flipper_length_ mm	body_mass_g	sex
count	344	344	342.000000	342.000000	342.000000	342.000000	334
unique	3	3	NaN	NaN	NaN	NaN	3
top	Adelie	Biscoe	NaN	NaN	NaN	NaN	MALE
freq	152	168	NaN	NaN	NaN	NaN	168
mean	NaN	NaN	43.921930	17.151170	200.915205	4201.754386	NaN
std	NaN	NaN	5.459584	1.974793	14.061714	801.954536	NaN
min	NaN	NaN	32.100000	13.100000	172.000000	2700.000000	NaN
25%	NaN	NaN	39.225000	15.600000	190.000000	3550.000000	NaN

	species	island	culmen_length_ mm	culmen_depth_ mm	flipper_length_ mm	body_mass_g	sex
50%	NaN	NaN	44.450000	17.300000	197.000000	4050.000000	NaN
75%	NaN	NaN	48.500000	18.700000	213.000000	4750.000000	NaN
max	NaN	NaN	59.600000	21.500000	231.000000	6300.000000	NaN

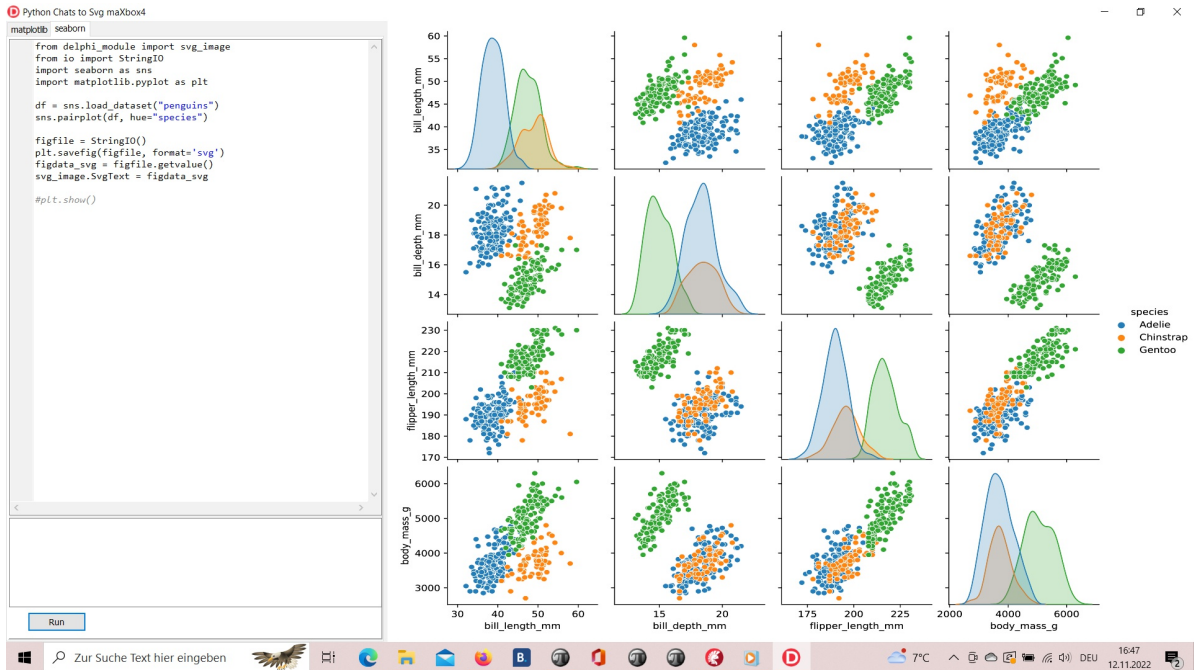


1096_2022-11-12_svg_python2seaborn.png

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   species                344 non-null    object
1   island                 344 non-null    object
2   culmen_length_mm      342 non-null    float64
3   culmen_depth_mm       342 non-null    float64
4   flipper_length_mm     342 non-null    float64
5   body_mass_g           342 non-null    float64
6   sex                    334 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
memory shape core cube #7
```

The dataset consists of 7 columns.

- **species:** penguin species (Chinstrap, Adélie, or Gentoo)
- **culmen_length_mm:** culmen length (mm)
- **culmen_depth_mm:** culmen depth (mm)
- **flipper_length_mm:** flipper length (mm)
- **body_mass_g:** body mass (g)
- **island:** island name (Dream, Torgersen, or Biscoe) in the Palmer Archipelago (Antarctica)
- **sex:** penguin sex



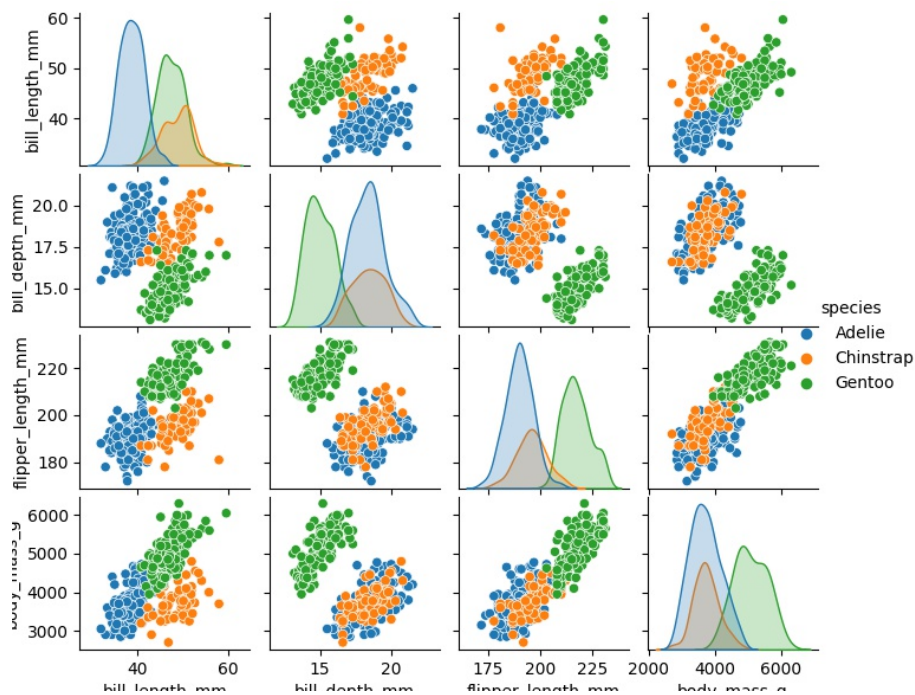
1096_2022-11-12_svg_python_delphi_scalable.png

Seaborn is a **Python** data visualization library based on **matplotlib**. It provides a high-level interface for drawing attractive and informative statistical graphics.

1.2 Covariance

“Covariance” indicates the direction of the linear relationship between variables. “Correlation” on the other hand measures both the strength and direction of the linear relationship between two variables.

Source: <https://tinyurl.com/yd2pezss>



1096_Figure_1_species.png

The scatter plot which shows us the correlation with respect to other features. This method helps just to figure out the important features which account the most for the classification in our model.

1.3 Scatter Plot

A scatter plot (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

A common modification of the basic scatter plot is the addition of a third variable. Values of the third variable can be encoded by modifying how the points are plotted. For a third variable that indicates categorical values (like geographical region or gender), the most common encoding is through point color.

Seaborn lets you create relational plots using the `relplot()` function. The function technically lets you create more than scatter plots.

Official releases of seaborn can be installed from [PyPI](#):

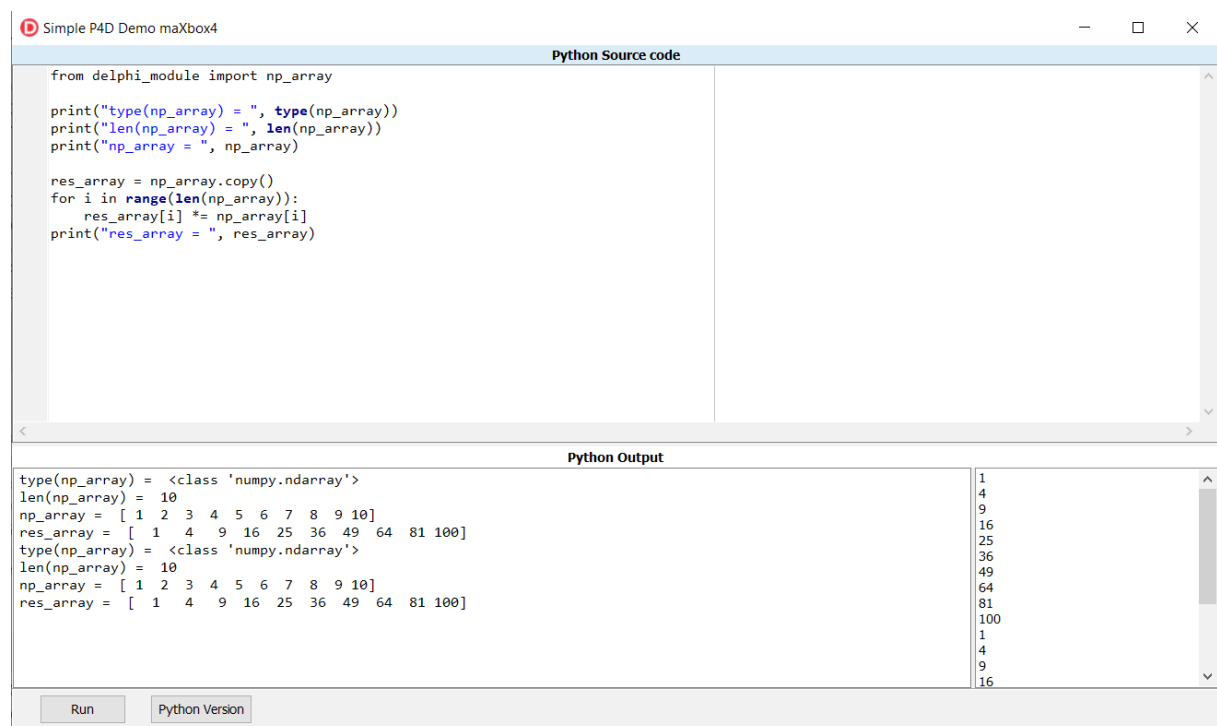
```
pip install seaborn
```

The basic invocation of `pip` will install seaborn and, if necessary, its mandatory dependencies. It is possible to include optional dependencies that give access to a few advanced features:

```
pip install seaborn[stats].
```

Numpy arrays are a good substitute for python lists. They are [better than python lists](#). They provide faster speed and take less memory space. Let's begin with its definition for those unaware of numpy arrays. They are multi-dimensional matrices or lists of fixed size with similar elements.

Pandas is a popular Python library used to manipulate tabular data. It provides a versatile `dataframe` object that can read data from many popular formats, such as Excel, SQL, CSV and more. The Pandas style API provides you with many different tools that makes working with styling tabular data much easier.



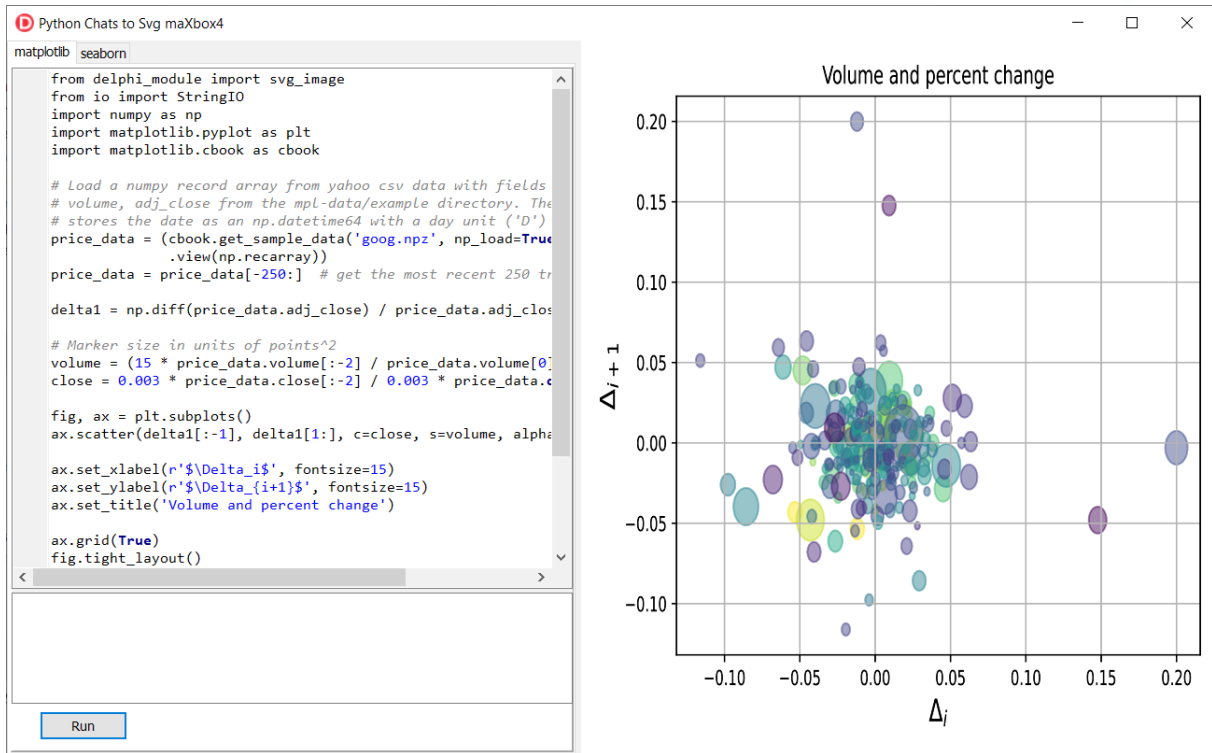
```
from delphi_module import np_array

print("type(np_array) = ", type(np_array))
print("len(np_array) = ", len(np_array))
print("np_array = ", np_array)

res_array = np_array.copy()
for i in range(len(np_array)):
    res_array[i] *= np_array[i]
print("res_array = ", res_array)
```

```
type(np_array) = <class 'numpy.ndarray'>
len(np_array) = 10
np_array = [ 1  2  3  4  5  6  7  8  9 10]
res_array = [ 1  4  9 16 25 36 49 64 81 100]
type(np_array) = <class 'numpy.ndarray'>
len(np_array) = 10
np_array = [ 1  2  3  4  5  6  7  8  9 10]
res_array = [ 1  4  9 16 25 36 49 64 81 100]
```

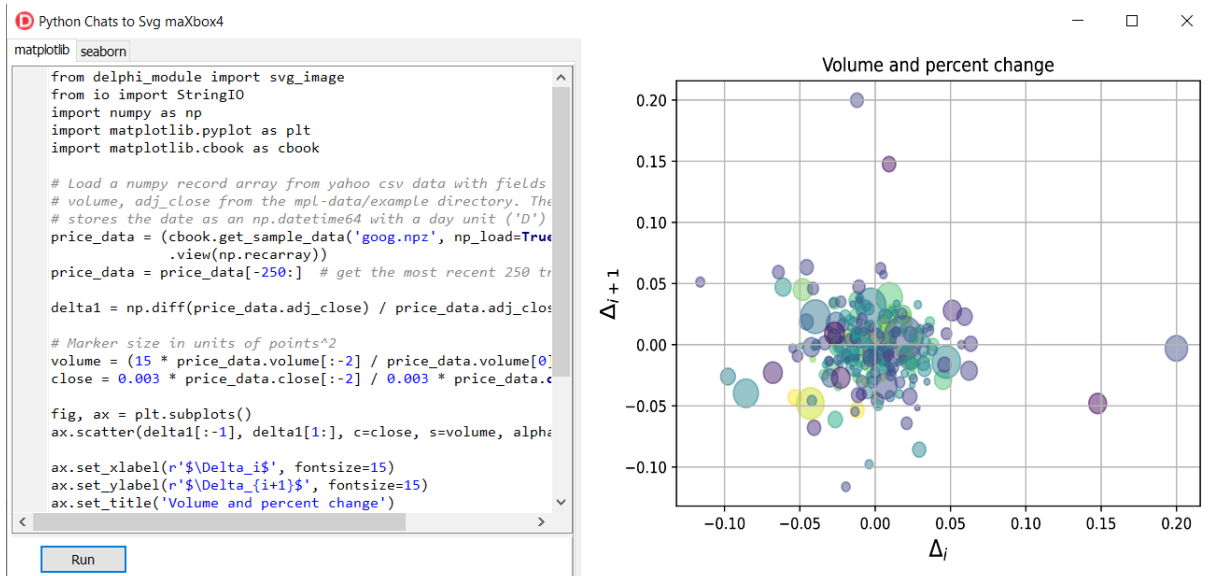
1096_2022-11-12_var_python.png



1096_2022-11-12_svg_python.png

python - Scaling a plot (matplotlib) - Stack Overflow

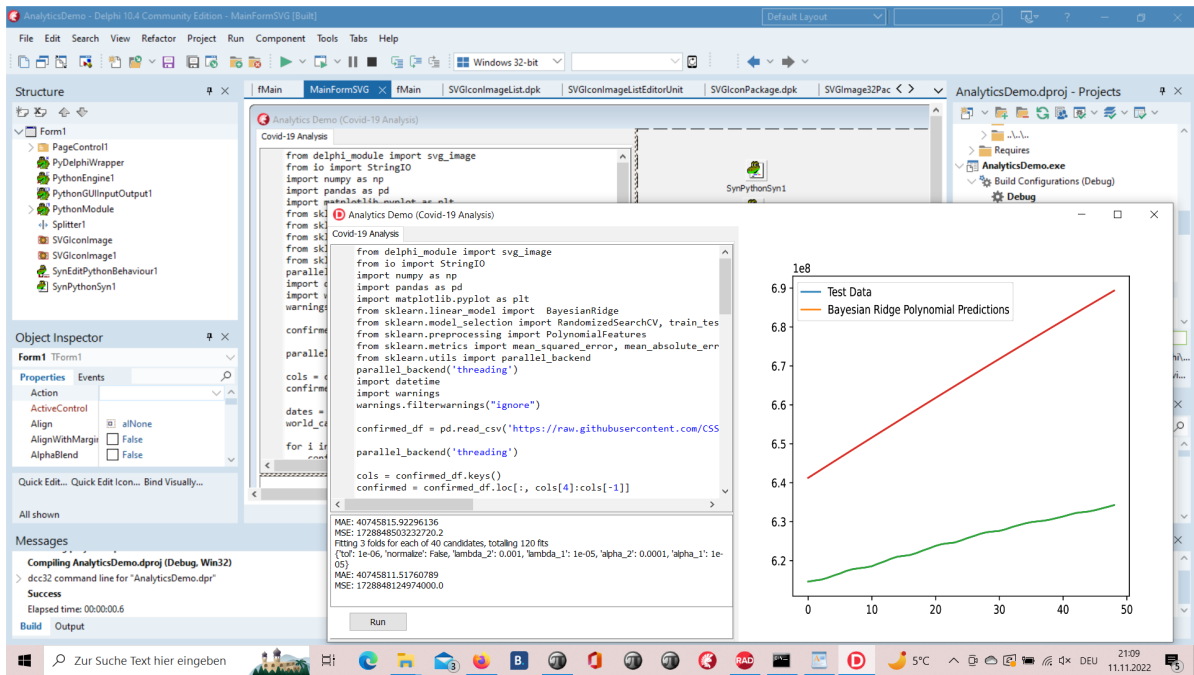
The graph is quite big, so in order to not only see many overlapping dots indicating the nodes, I have to scale the output picture. I used: `f,ax = plt.subplots(1,1)` `ax.set_aspect('equal')` `zoom=30` `w, h = f.get_size_inches()` `f.set_size_inches(w * zoom, h * zoom)`.



1096_2022-11-12_svg_python2.png

1.4 Predict and Dict the Pic

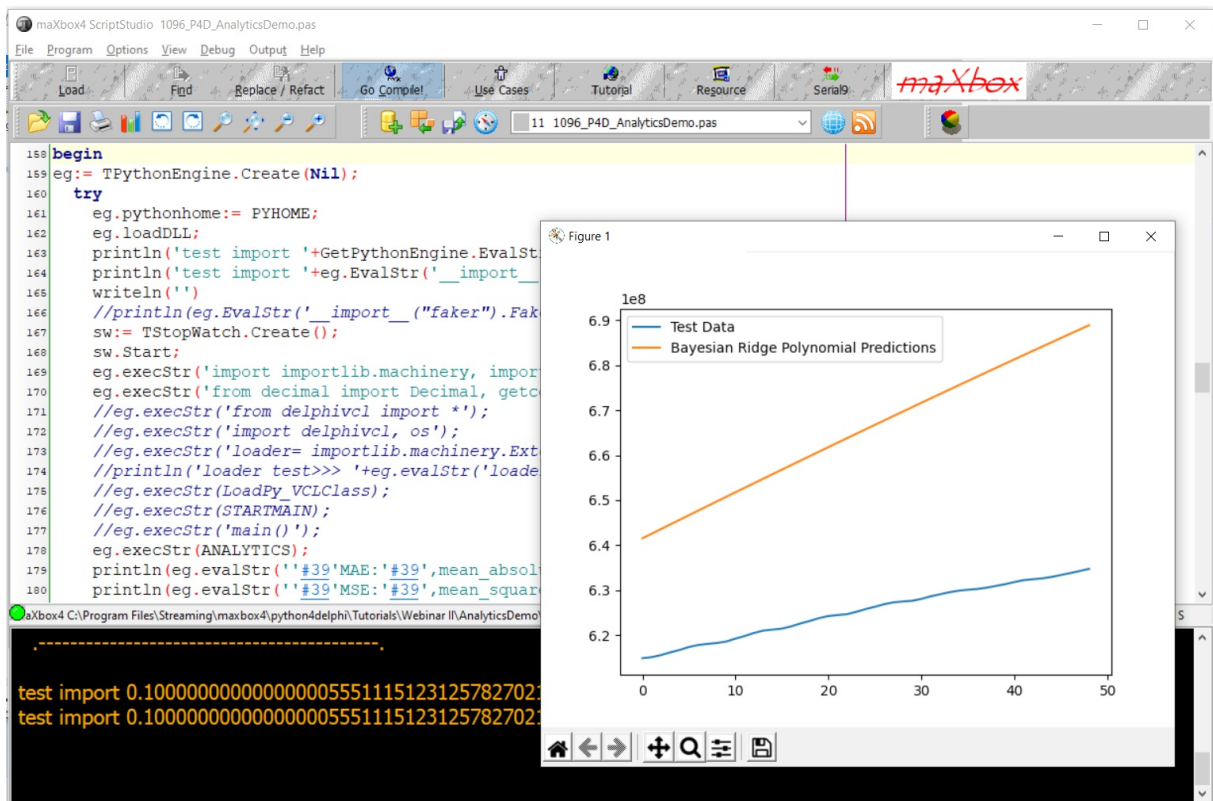
When you hear the word, 'Bayesian', you might think of Naive Bayes. However, Bayesian principles can also be used to perform regression. In this article, we will discuss and implement **Bayesian Ridge Regression**, which is not the same as regular Ridge Regression. To understand more about regular Ridge Regression, you can follow [this](#) link.



1096_2022-11-11_analytics.png

1.5 Predict in a Script

Regression is a Machine Learning task to predict continuous values (real numbers), as compared to classification, that is used to predict categorical (discrete) values.



1096_2022-11-11_analytics_script.png

The algorithm uses a hyper parameter to control regularization strength and fully integrates over the hyper-parameter in the posterior distribution, applying a hyper-prior selected so as to be approximately non-informative.

Scikit-learn's algorithm makes use of conjugate priors and because of that is restricted to use the Gamma prior which requires four hyper-parameters chosen arbitrarily to be small values. Additionally, it requires initial values for parameters α and λ that are then updated from the data.

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import BayesianRidge
```

In comparison, the algorithm we presented requires no initial parameters; and because the hyperparameter is integrated over, poor performing values contribute little to the posterior probability mass.

```
#####
# Plot the true and predicted curves for bbai's BayesianRidgeRegression
model
from bbai.glm import BayesianRidgeRegression
reg = BayesianRidgeRegression(fit_intercept=False)
fig, ax = plt.subplots(1, 1, figsize=(4, 4))

# Note: there are no parameters to tweak
reg.fit(X_train, y_train)
ymean, ystd = reg.predict(X_test, return_std=True)

ax.plot(x_test, func(x_test), color="blue", label="sin($2\pi x$)")
ax.scatter(x_train, y_train, s=50, alpha=0.5, label="observation")
ax.plot(x_test, ymean, color="red", label="predict mean")
ax.fill_between(
    x_test, ymean - ystd, ymean + ystd, color="pink", alpha=0.5,
    label="predict std"
)
ax.set_ylim(-1.3, 1.3)
ax.legend()

plt.tight_layout()
plt.show()
```

2 Classify

Since the missing values are already we don't need to worry about that. Next step is to encode the categorical variables.

I am setting 'sex' as the target variable. So the categorical variables to be encoded are 'species' and 'island'.

```
df = data.copy()
target = 'sex'
encode = ['species', 'island']

for col in encode:
    dummy = pd.get_dummies(df[col], prefix=col)
    df = pd.concat([df, dummy], axis=1)
    del df[col]
```

Lets label encode the target variable as well.

I won't be using any fit transform from the scikit learn api rather i will be using a primitive mapping. Otherwise you get :

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
File"C:\Users\breitsch\AppData\Local\Programs\Python\Python38\lib\site-  
packages\pandas\core\generic.py",line1778,in __array__  
returnnp.asarray(self._values,dtype=dtype)  
File"C:\Users\breitsch\AppData\Local\Programs\Python\Python38\lib\site-  
packages\numpy\core\_asarray.py",line83,inasarray  
returnarray(a,dtype,copy=False,order=order)  
ValueError:couldnotconvertstringtofloat:'Gentoo'
```

```
target_mapper = {'MALE':0, 'FEMALE':1}  
def target_encode(val):  
    return target_mapper[val]  
  
df['sex'] = df['sex'].apply(target_encode)
```

separating X and y

```
X = df.drop('sex', axis=1)  
y = df['sex']
```

scaling the data

```
from sklearn import preprocessing  
X = preprocessing.scale(X)
```

splitting the data

```
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=13)
```

model fitting and prediction

```
from sklearn.linear_model import LogisticRegression  
  
model = LogisticRegression().fit(X_train, y_train)  
pred = model.predict(X_test)
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
(344,10)  
(344,)
```

```
Traceback(mostrecentcalllast):
```

```
File"C:\ProgramFiles\Streaming\IBZ2021\Module2_3\992_oma_objectdetector21py  
scripter.py",line133,in<module>  
X[:,~np.isnan(X).any(axis=1)]  
IndexError:booleanindexdidnotmatchindexedarrayalongdimension1;dimensionis10  
butcorrespondingbooleandimensionis344
```

When using a dataset for analysis, you must check your data to ensure it only contains finite numbers and no NaN values (Not a Number). If you try to pass a dataset that contains NaN or infinity values to a function for analysis, you will raise the error: `ValueError: input contains nan, infinity or a value too large for dtype('float64')`.

```
df=df.dropna()
```

```
from sklearn.metrics import classification_report, confusion_matrix,
roc_curve, roc_auc_score
```

```
print('CONFUSION MATRIX')
print(confusion_matrix(y_test, pred))
```

CONFUSION MATRIX

```
[[29  7]
 [ 1 32]]
```

```
print('CLASSIFICATION REPORT\n')
print(classification_report(y_test, pred))
```

CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.97	0.81	0.88	36
1	0.82	0.97	0.89	33
accuracy			0.88	69
macro avg	0.89	0.89	0.88	69
weighted avg	0.90	0.88	0.88	69

Now I am setting '**species**' as the target variable. So the categorical variables to be encoded are no values because I do have numerical values. Latter we add one categorical value as a important feature to classify to 100%!

```
X=df[["bill_length_mm", "bill_depth_mm", "flipper_length_mm"]]
y=df["species"]
```

```
print(X.shape)
print(y.shape)
```

```
(266, 3)
(266,)
```

We do have 266 samples with 3 features to classify with a logistic regression:

```
from sklearn import preprocessing
X=preprocessing.scale(X)
#splitting the data
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=13)
#model fitting and prediction
from sklearn.linear_model import LogisticRegression

model = LogisticRegression().fit(X_train,y_train)
pred = model.predict(X_test)
print(pred)
```

#now checking performance of model

```

from sklearn.metrics import
    classification_report, confusion_matrix, roc_curve, roc_auc_score

print('CONFUSIONMATRIX')
print(confusion_matrix(y_test, pred))
print('CLASSIFICATIONREPORT\n')
print(classification_report(y_test, pred))

```

```

['Gentoo' 'Chinstrap' 'Adelie' 'Adelie' 'Gentoo' 'Gentoo' 'Adelie' 'Adelie'
'Adelie' 'Adelie' 'Gentoo' 'Adelie' 'Adelie' 'Adelie' 'Chinstrap' 'Gentoo'
Chinstrap' 'Gentoo' 'Gentoo' 'Gentoo' 'Adelie' 'Adelie' 'Adelie' 'Chinstrap'
'Gentoo' 'Chinstrap' 'Adelie' 'Adelie' 'Gentoo' 'Chinstrap' 'Gentoo'
'Chinstrap' 'Adelie' 'Gentoo' 'Gentoo' 'Adelie' 'Gentoo' 'Chinstrap'
'Adelie' 'Chinstrap' 'Adelie' 'Adelie' 'Gentoo' 'Adelie' 'Adelie' 'Adelie'
'Gentoo' 'Adelie' 'Gentoo' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
'Gentoo' 'Chinstrap' 'Chinstrap' 'Adelie' 'Gentoo' 'Adelie' 'Chinstrap'
'Adelie' 'Gentoo' 'Gentoo' 'Gentoo' 'Adelie' 'Adelie']

```

CONFUSION MATRIX

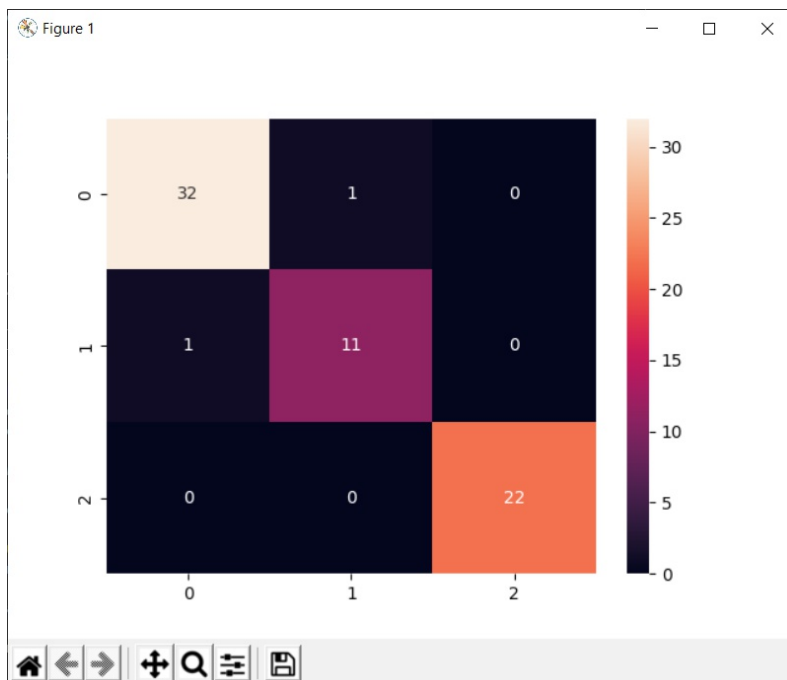
```
[[32  1  0]  33
```

```
 [ 1 11  0]  12
```

```
 [ 0  0 22]] 22
```

CLASSIFICATION REPORT

	precision	recall	f1-score	support
Adelie	0.97	0.97	0.97	33
Chinstrap	0.92	0.92	0.92	12
Gentoo	1.00	1.00	1.00	22
accuracy			0.97	67
macro avg	0.96	0.96	0.96	67
weighted avg	0.97	0.97	0.97	67

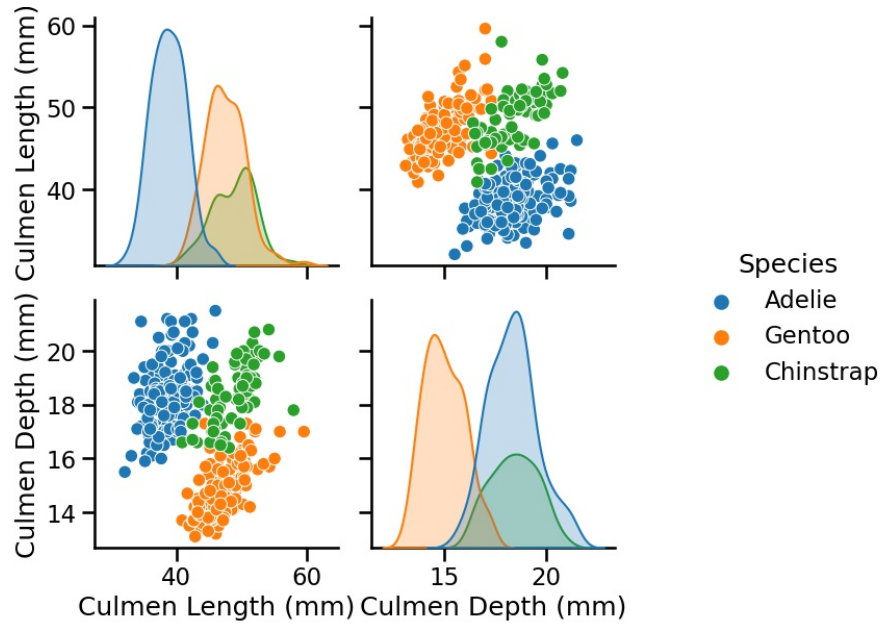


```
>>> >>> 1096_Figure_1_species2confusionmatrix.png
```

So we have one false positive and one false negative between Adelie and Chinstrap. Gentoo has no error, but why ?

<https://datatofish.com/confusion-matrix-python/>

The Gentoo penguin is a penguin species in the genus *Pygoscelis*, most closely related to the Adélie penguin and the chinstrap penguin. As we have seen each penguin is from one of the three following species: Adelie, Gentoo, and Chinstrap. See the illustration above depicting in a pair plot correlation matrix the three different penguin species. But the Gentoo is simple to recover as a almost clear distinction.

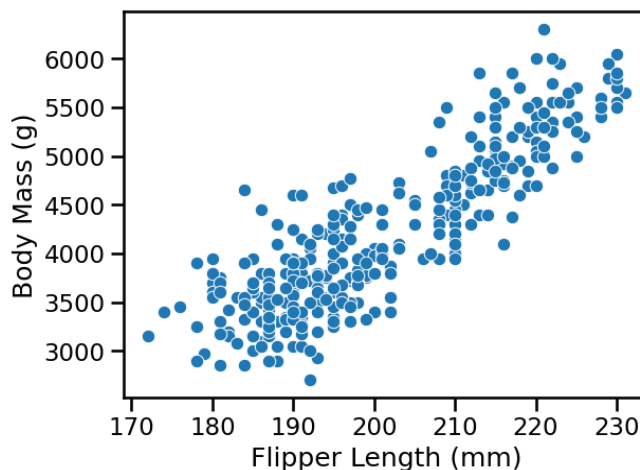


1096_trees_dataset_5_0_culmen.png

We can deduce the following intuitions:

- The Adelie species can be differentiated from the Gentoo and Chinstrap species depending on the culmen length;
- The Gentoo species can be differentiated from the Adelie and Chinstrap species depending on the culmen depth.

From the scatter plot below, we observe that we have a linear relationship between the flipper length and the body mass. The longer the flipper of a penguin, the heavier the penguin.



1096_trees_dataset_5_0_culmen_bodymass.png

So we have the `flipper_length_mm` as a feature . We can also use this data-set for both classification and regression problems by selecting a subset of the features to make our explanations intuitive. Another information is the island , cause certain species have their own habitat. If we add a specific island as a forth feature we do made a cheap trick to get a 100 % classification:

```
encode = ['island']

for col in encode:
    dummy = pd.get_dummies(df[col], prefix=col)
    df = pd.concat([df,dummy], axis=1)
    del df[col]

print(dummy)
```

dtype: int64

```
island_Biscoe island_Dream island_Torgersen
0          0          0          1
1          0          0          1
2          0          0          1
4          0          0          1
5          0          0          1
..         ...          ...          ...
```

```
X = df[["bill_length_mm", "bill_depth_mm", "flipper_length_mm", "island_Dream"]]
```

Indeed, we will use features based on penguins' bill and flipper measurement, but we also checked with culmen and only culmen.

```
culmen_columns = ["Culmen Length (mm)", "Culmen Depth (mm)"]
```

<https://allisonhorst.github.io/palmerpenguins/>

That was a joke because culmen and bill are the same!

The culmen is the upper ridge of a bird's bill. In the simplified penguins data, culmen length and depth are renamed as variables `bill_length_mm` and `bill_depth_mm` to be more intuitive. For this penguin data, the culmen (bill) length and depth are measured.

After encoding a categorical value we set `island_Dream` in our features and got 100 % :

CONFUSION MATRIX

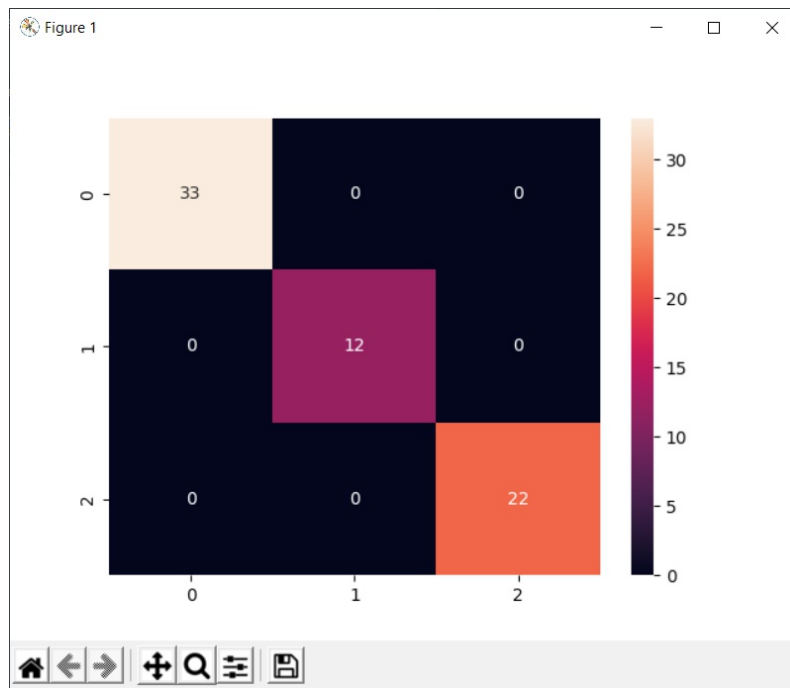
```
[[33 0 0]
```

```
[ 0 12 0]
```

```
[ 0 0 22]]
```

CLASSIFICATION REPORT

	precision	recall	f1-score	support
Adelie	1.00	1.00	1.00	33
Chinstrap	1.00	1.00	1.00	12
Gentoo	1.00	1.00	1.00	22
accuracy			1.00	67
macro avg	1.00	1.00	1.00	67
weighted avg	1.00	1.00	1.00	67



1096_Figure_1_species2confusionmatrix2.png

You can use the seaborn package in Python to get a more vivid display of the matrix. To accomplish this task, you'll need to add the following two components into the code:

- `import seaborn as sn`
- `sn.heatmap(confusion_matrix, annot=True)`

A last test is the ROC: # ROC CURVE

```
print('ROC CURVE')
train_probs = model.predict_proba(X_train)
train_probs1 = train_probs[:, 1]
fpr0, tpr0, thresholds0 = roc_curve(y_train, train_probs1)

test_probs = model.predict_proba(X_test)
test_probs1 = test_probs[:, 1]
fpr1, tpr1, thresholds1 = roc_curve(y_test, test_probs1)

plt.plot(fpr0, tpr0, marker='.', label='train')
plt.plot(fpr1, tpr1, marker='.', label='validation')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

But we got an error:

ValueError: multiclass format is not supported

Thats clear cause we have 3 classes instead of 2 like sex is:

So we drop one class:

```
df=df.drop(df[df['species']=='Gentoo'].index)
```

then we got in ROC:

```
raise ValueError("y_true takes value in {{{classes_repr}}} and "
```

```
ValueError: y_true takes value in {'Adelie', 'Chinstrap'} and pos_label is not specified: either make y_true take value in {0, 1} or {-1, 1} or pass pos_label explicitly.
```

```
from collections import Counter
```

```
print(Counter(y_train)) # y_true must be your labels
```

```
print('ROC CURVE')
```

```
train_probs = model.predict_proba(X_train)
```

```
train_probs1 = train_probs[:, 0]
```

```
fpr0,tpr0,thresholds0=roc_curve(y_train,train_probs1,pos_label='Adelie')
```

```
test_probs = model.predict_proba(X_test)
```

```
test_probs1 = test_probs[:, 0]
```

```
fpr1,tpr1,thresholds1=roc_curve(y_test,test_probs1,pos_label='Adelie')
```

```
plt.plot(fpr0, tpr0, marker='.', label='train')
plt.plot(fpr1, tpr1, marker='.', label='validation')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
>>> CONFUSION MATRIX
```

```
[[71  1]
```

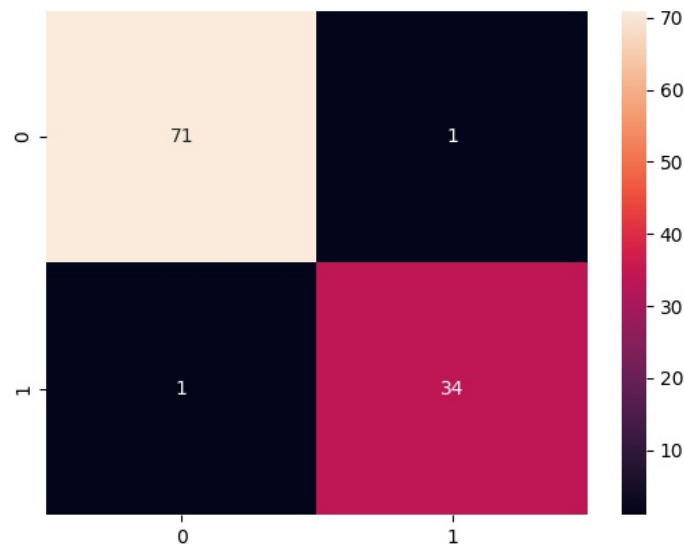
```
 [ 1 34]]
```

	precision	recall	f1-score	support
Adelie	0.99	0.99	0.99	72
Chinstrap	0.97	0.97	0.97	35

```
Counter({'Adelie': 74, 'Chinstrap': 33})
```

ROC CURVE and CM

Compute Receiver operating characteristic (ROC).



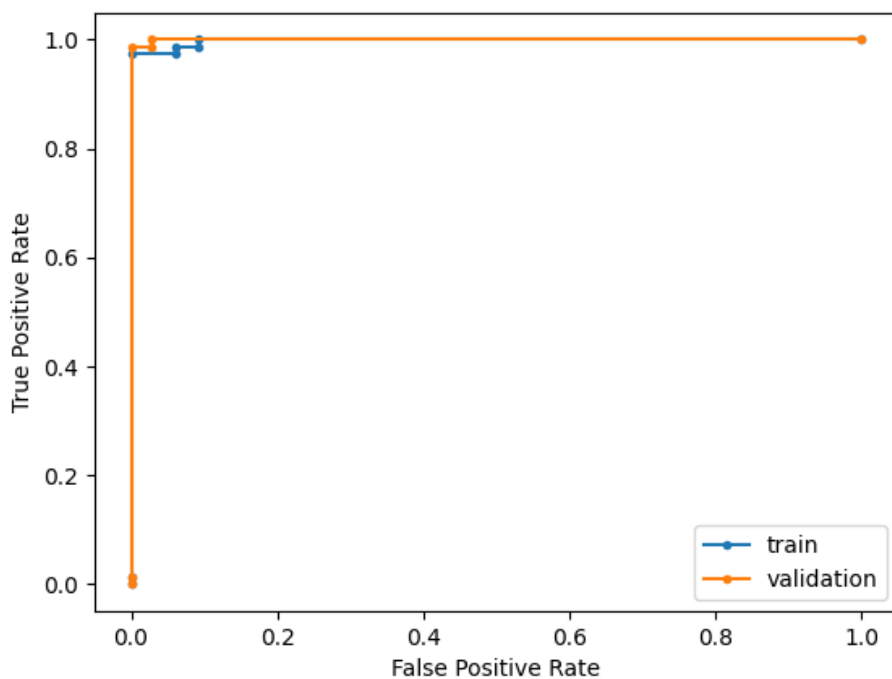
1096_Figure_1_species2confusionmatrix3.png

This means that all the values in `y_true` are 34 or 71, which means there is positive class records in the given dataset.

Instead of passing 2 arguments in your function, pass an additional parameter stating the `pos_label` just like the error states while elaborating.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

`fpr0, tpr0, thresholds0=roc_curve(y_train, train_probs1, pos_label='Adelie')`



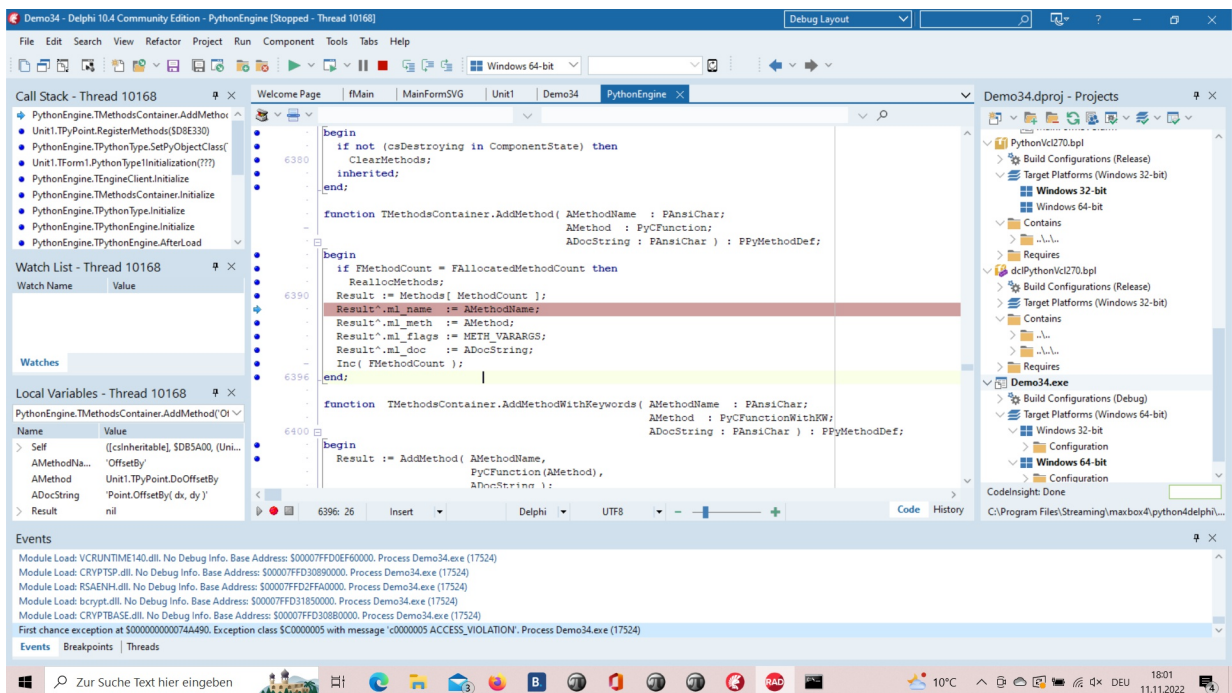
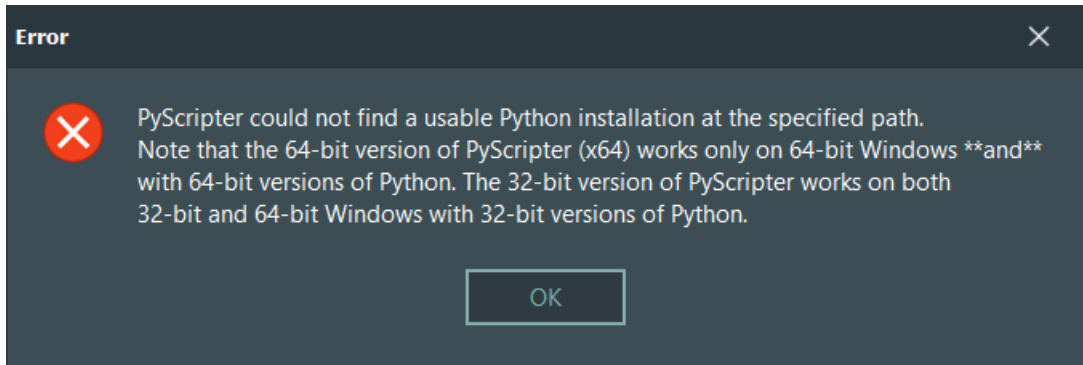
1096_Figure_1_species2roc3.png

3 Technical Description

[maxkleiner \(Max Kleiner\) \(github.com\)](https://github.com/maxkleiner)

A Python program terminates as soon as it encounters an error. In Python, an error can be a syntax error or an exception. In his article, you will see what an exception is and how it differs from a syntax error. After that, you will learn about raising exceptions and making assertions. Then, you'll finish with a demonstration of the try and except block.

<https://realpython.com/python-exceptions/>

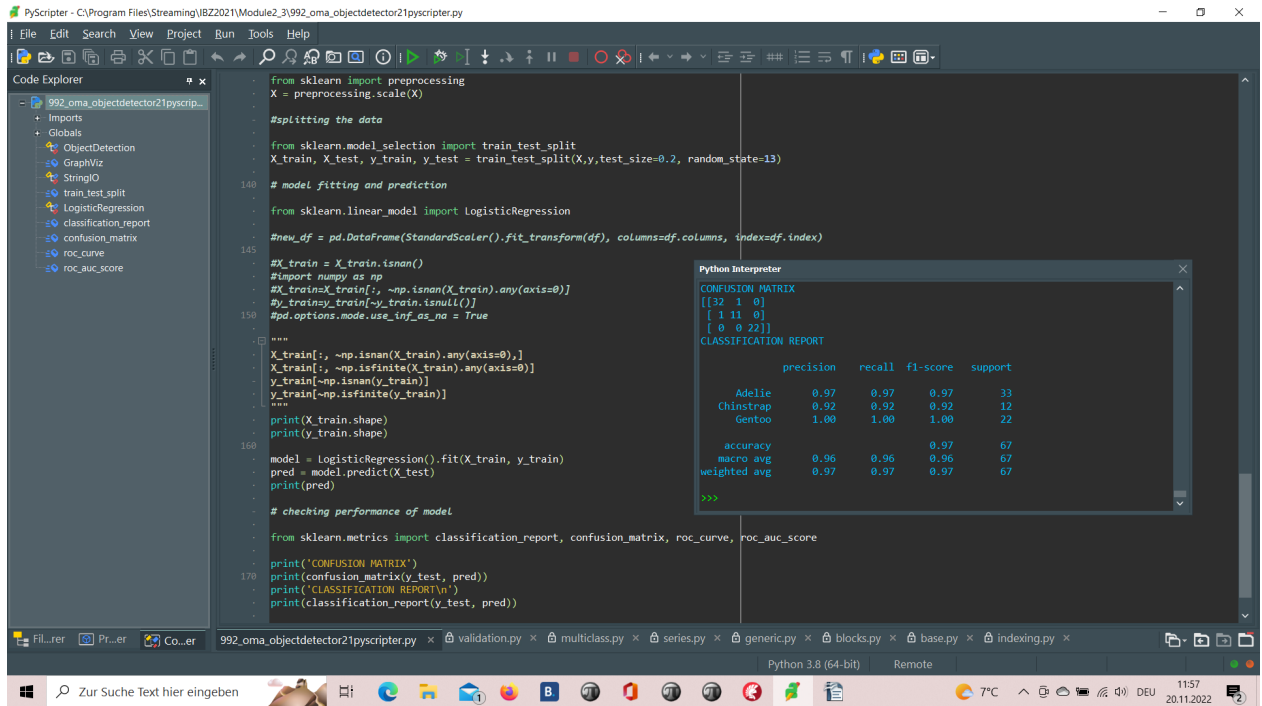


This Exception was prior to change between 32- and 64 bit.

1096_2022-11-11_exception.png

PyScripter 4.0.0 is now available at Sourceforge. This is a major new release with an updated User Interface and many significant enhancements under the hood, that will increase the stability and improve the user experience.

Sick of using IDLE? Want to code in a fancy new IDE? These steps will help you in downloading Python 2.7.1 or Python 3.8 and PyScripter so you can start learning and have fun with Python.

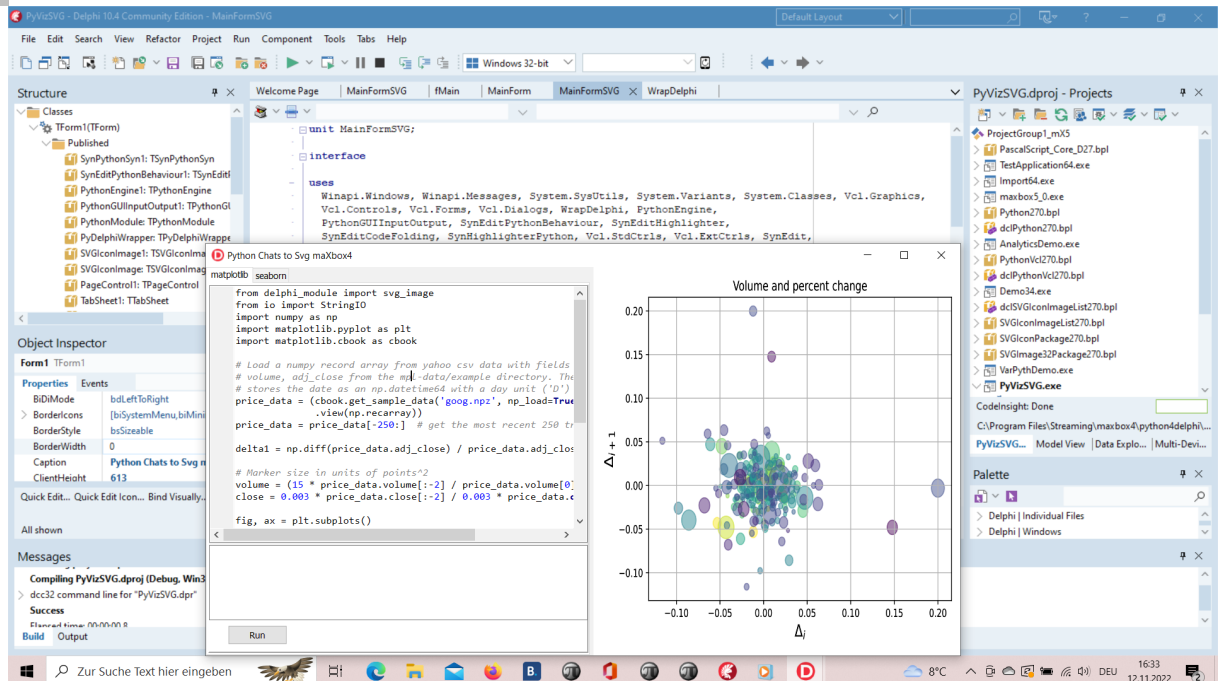


1096_2022-11-12_develop_overview2.png

<https://www.instructables.com/Python-With-PyScripter/>

4 Develop Overview

We value the quality of the material each student is learning. Our Instructor led classes help provide hands-on learning, one-on-one mentoring with experienced developers and peer-to-peer learning. To allow the most flexibility for students, we have Day and Night programs to choose from. *No prior coding experience required.



1096_2022-11-12_develop_overview.png

procedure pyBank_VCL4Python;

```
var eg: TPythonEngine; sw: TStopWatch;
begin
  eg:= TPythonEngine.Create (Nil);
  try
    eg.pythonhome:= PYHOME;
    eg.loadDLL;
    println('test import
'+GetPythonEngine.EvalStr('__import__("decimal").Decimal(0.1)'));
    writeln('')
    //println(eg.EvalStr('__import__("faker").Faker()'));
    sw:= TStopWatch.Create();
    sw.Start;
    eg.execStr('import importlib.machinery, importlib.util');
    eg.execStr('from decimal import Decimal, getcontext');
    importlib.machinery.ExtensionFileLoader("DelphiVCL", '+VCLHOME+')

    eg.execStr(ANALYTICSSVG);

//println(eg.evalStr('#39'MSE:'#39',mean_squared_error(test_bayesian_pred,
y_test_confirmed)'));

    sw.Stop;
    //sw.ElapsedMilliseconds;
    writeln('Stop Analytics Tester1: '+sw.getValueStr)
  except
    eg.raiseError;
    writeln(ExceptionToString(ExceptionType, ExceptionParam));
  finally
    eg.Free;
    sw.Free;
    sw:= Nil;
  end;
end;

print('ROC CURVE')

train_probs = model.predict_proba(X_train)

train_probs1 = train_probs[:, 0]

fpr0,tpr0,thresholds0 = roc_curve(y_train,train_probs1,pos_label='Adelie')

test_probs = model.predict_proba(X_test)

test_probs1 = test_probs[:, 0]

fpr1,tpr1,thresholds1 = roc_curve(y_test,test_probs1,pos_label='Adelie')

plt.plot(fpr0, tpr0, marker='.', label='train')

plt.plot(fpr1, tpr1, marker='.', label='validation')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.legend()

plt.show()
```