Python Training and Predictive Modelling (NLP)

Digital Accelerator
October 2019

pwc

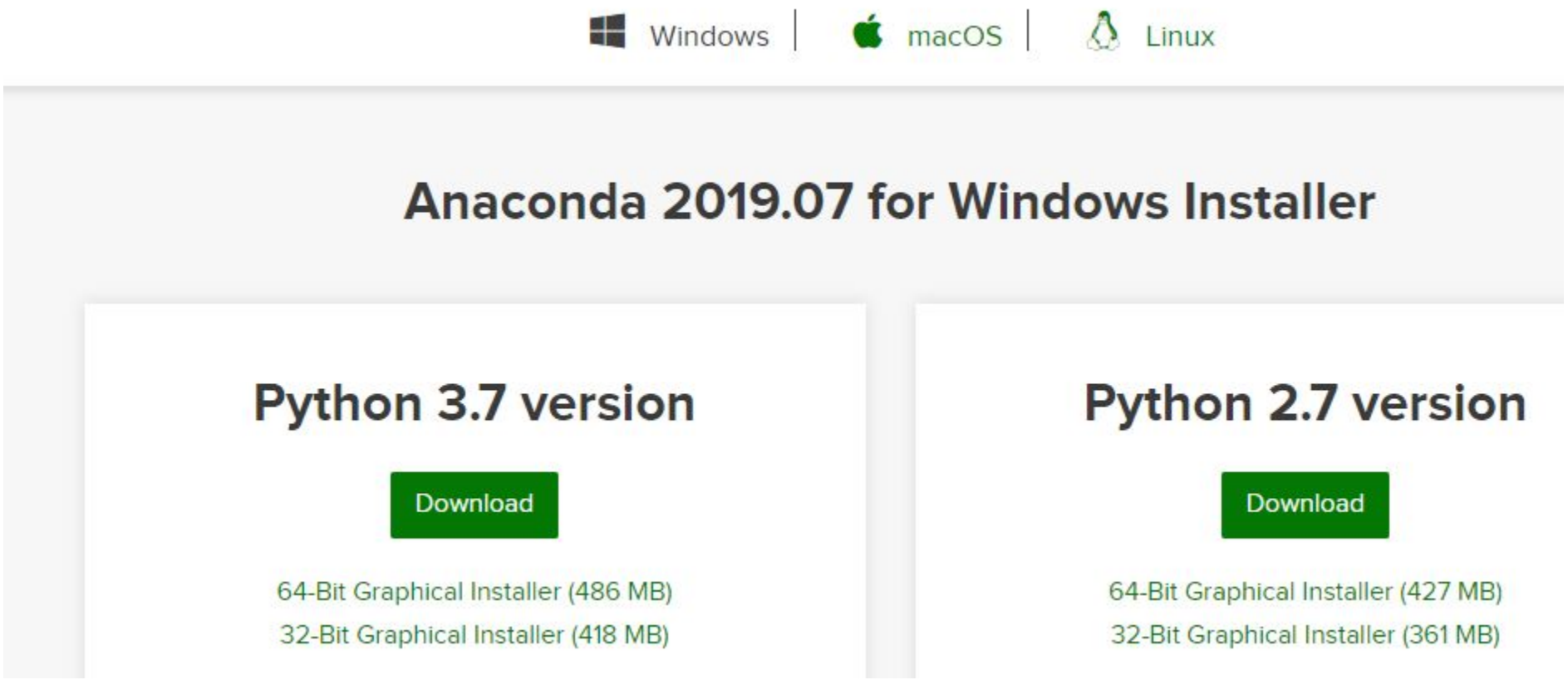
Agenda

1. An Introduction to Python programming 00
2. Popular Use Cases 00
3 Cleaning your data 00
4. Transforming categorical variables 00
5. Descriptive Statistics 00
6. Text Mining 00
7. Predictive Modelling 00
8. Data Visualization
9. Turning your code into a client presentation
10. Equivalent code in R programming

Pre-Work: Installing Python 3.7

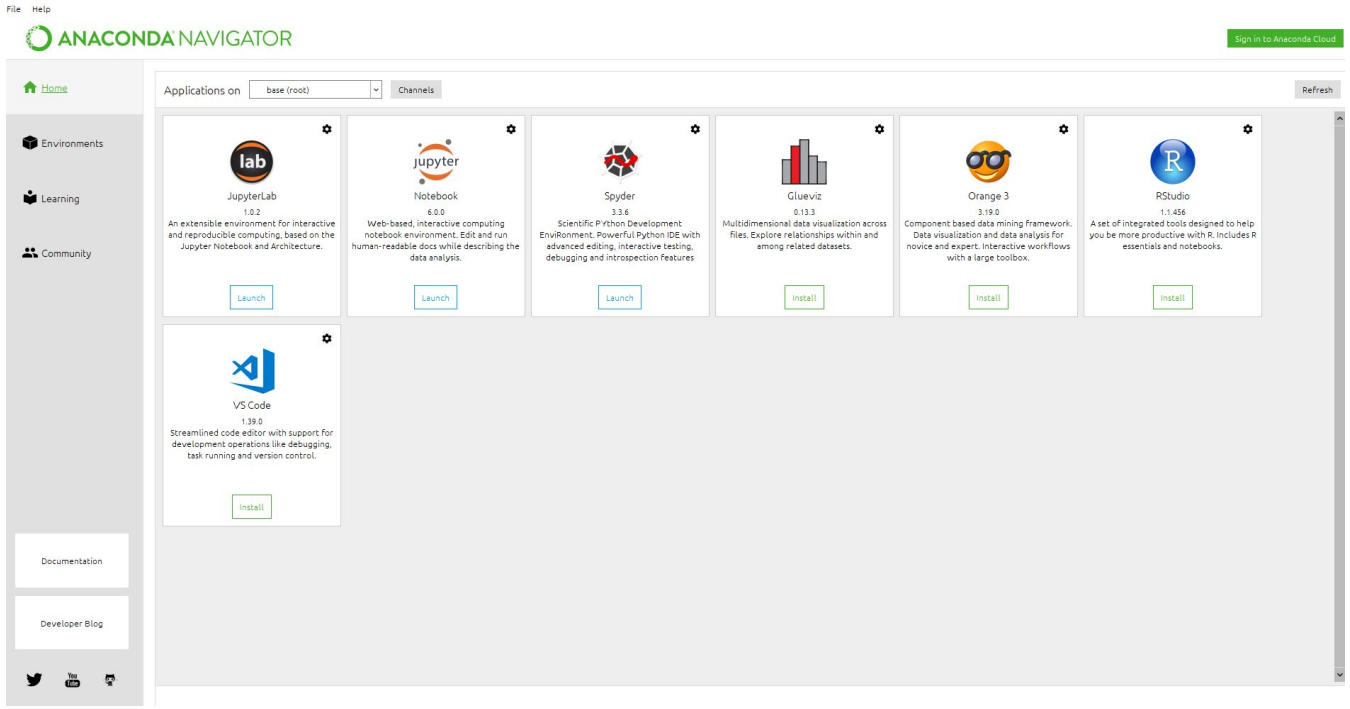
Download Python 3.7 version

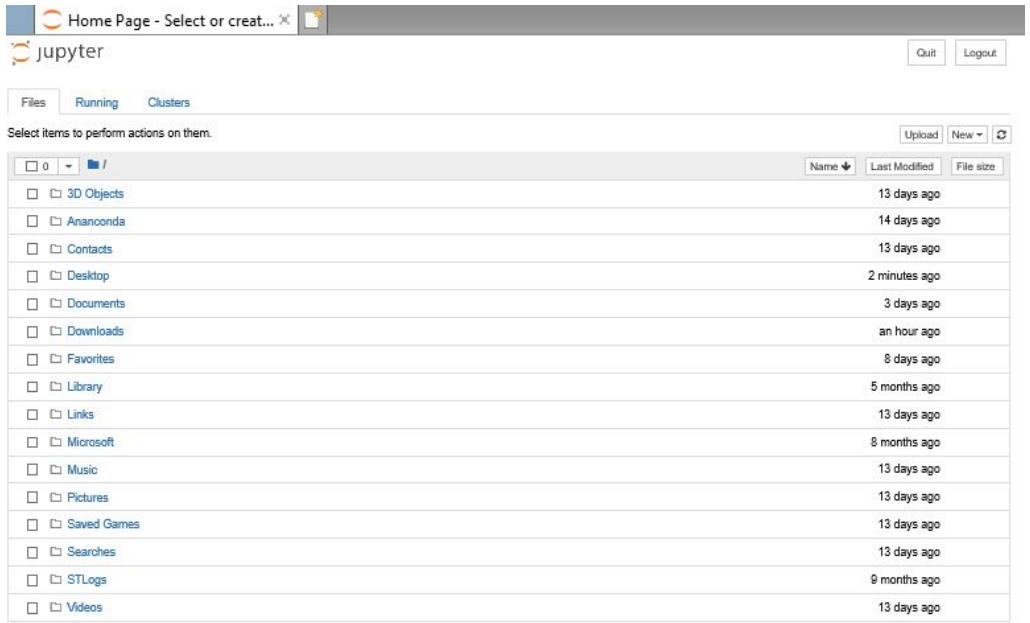
https://www.anaconda.com/distribution/



Launch Python 3.7 from Windows

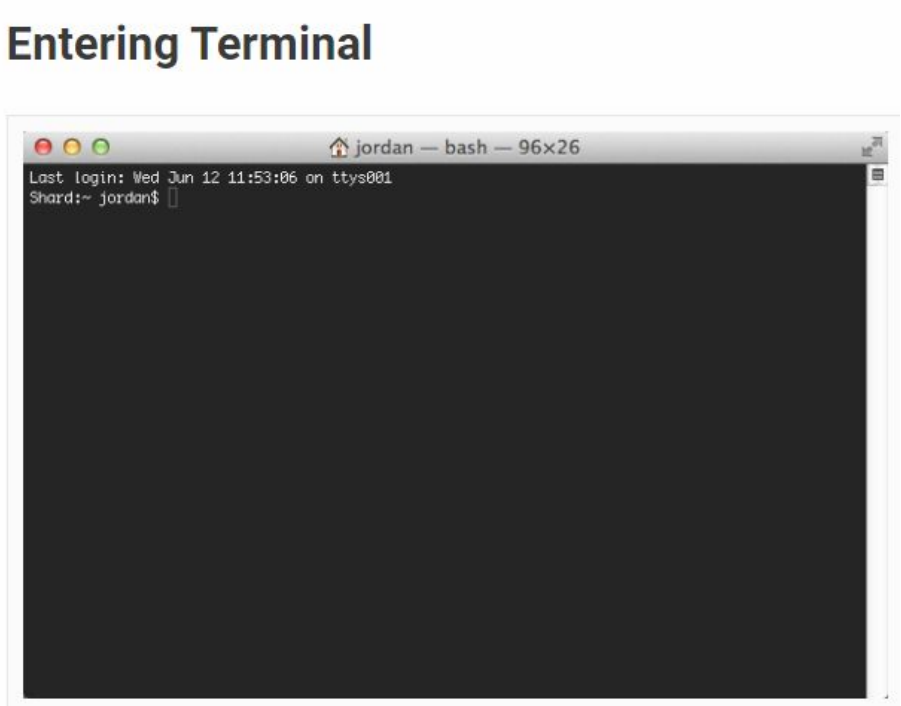
1. Create a **new folder** e.g. Python Training from your Desktop.
2. From your **Start Menu**, type **Anaconda Navigator (Anaconda)**.
3. Click **Ok** on the pop up message.
4. Click **Launch** from **Jupyter Notebook.**

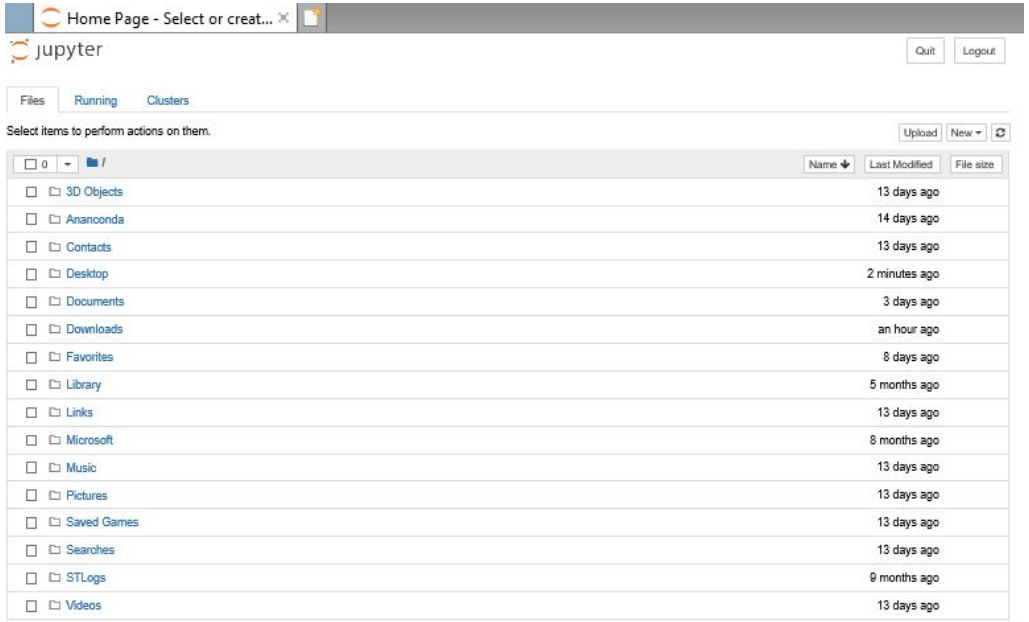




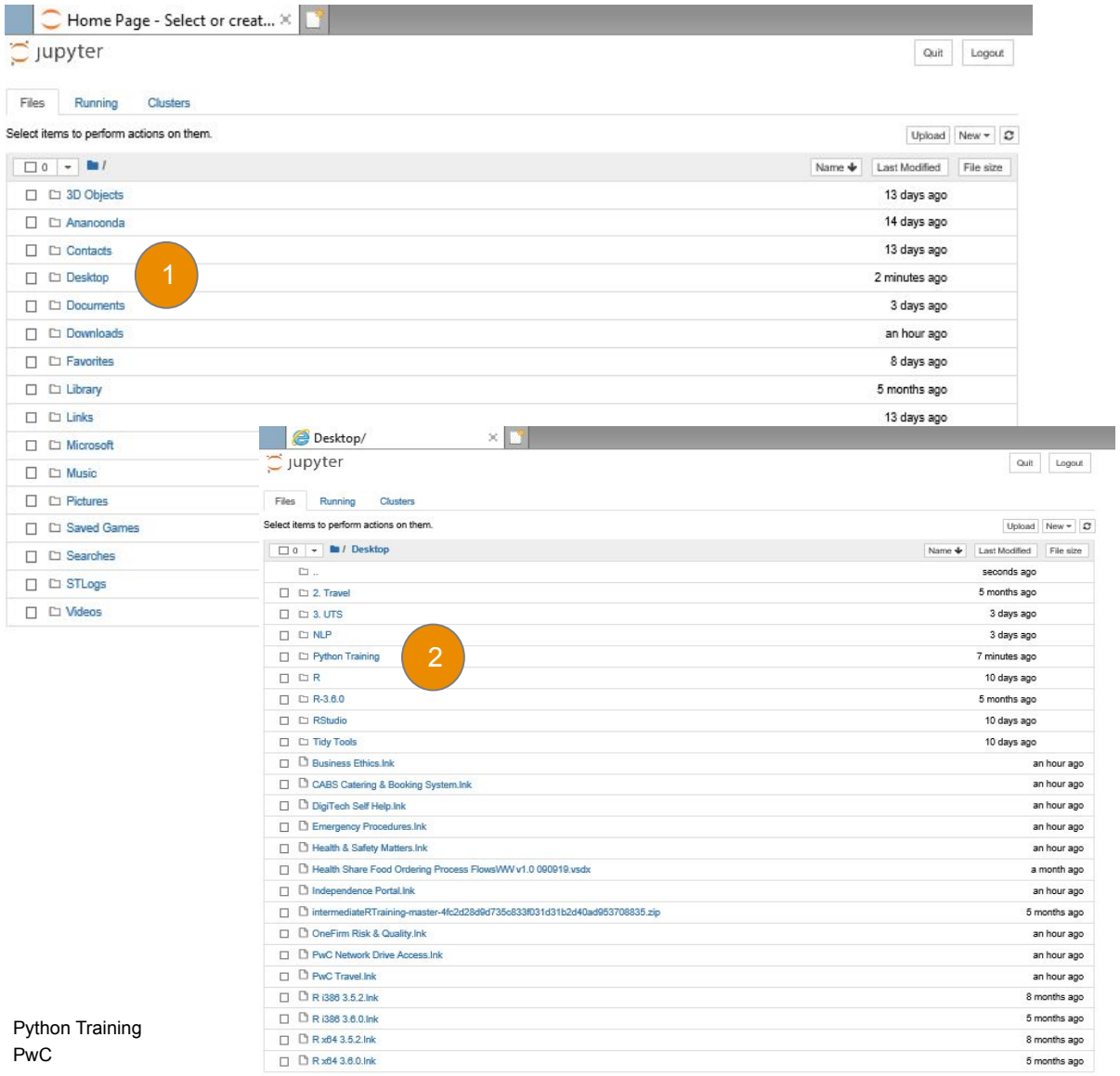
Launch Python 3.7 from Mac

1. Create a **new folder** e.g. Python Training from your Desktop.
2. In **Terminal**, type **Jupyter Notebook.**

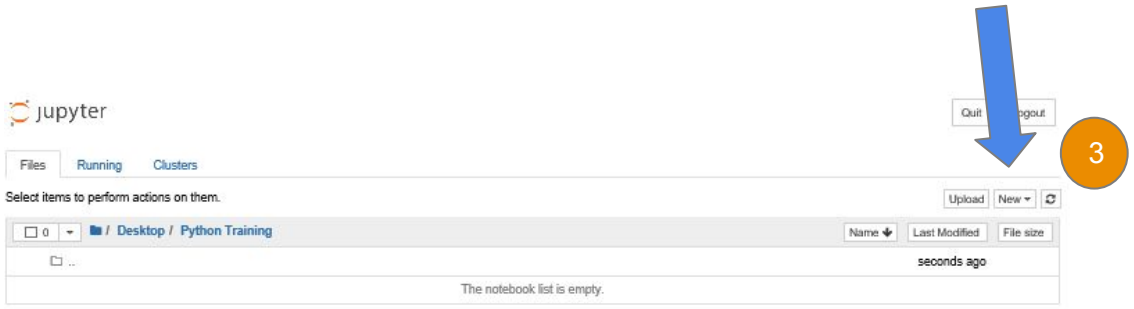




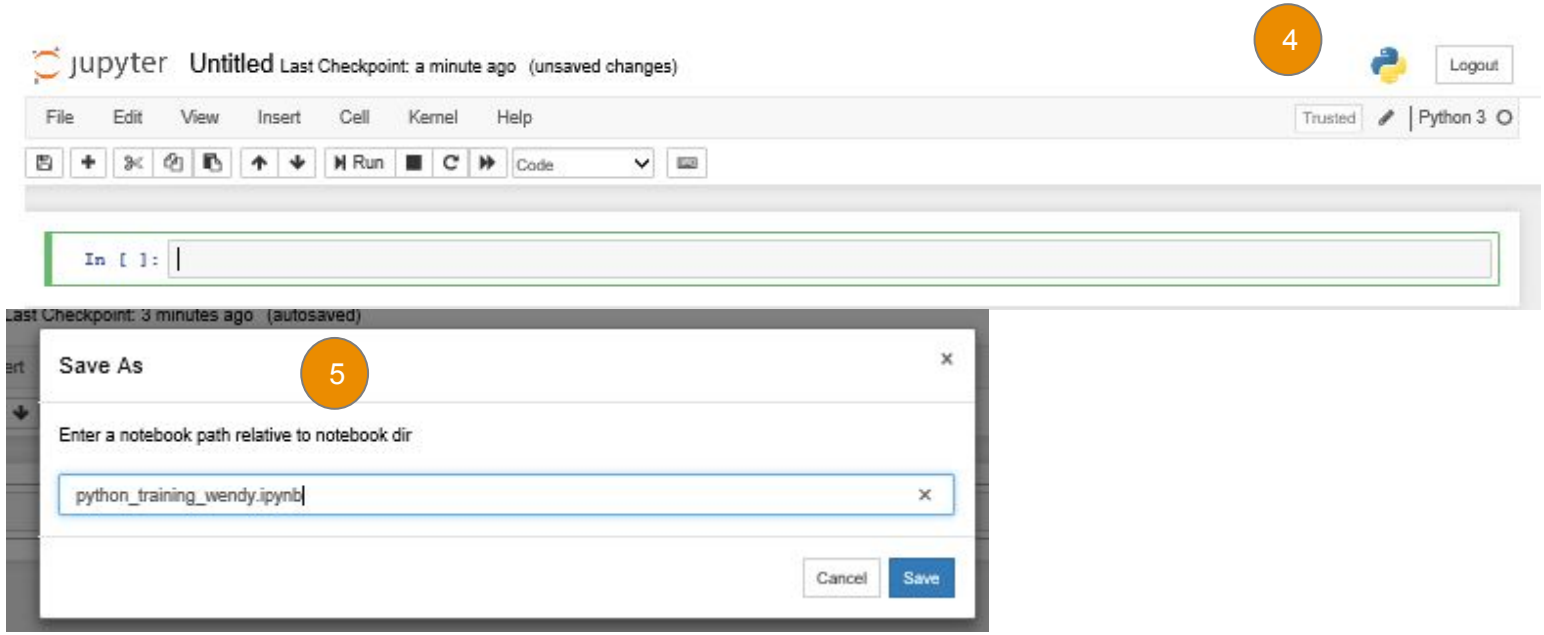
Jupyter Notebook



1. Navigate to **Desktop.**
2. Locate the folder **Python Training.**
3. On the far-right side, click **New.**



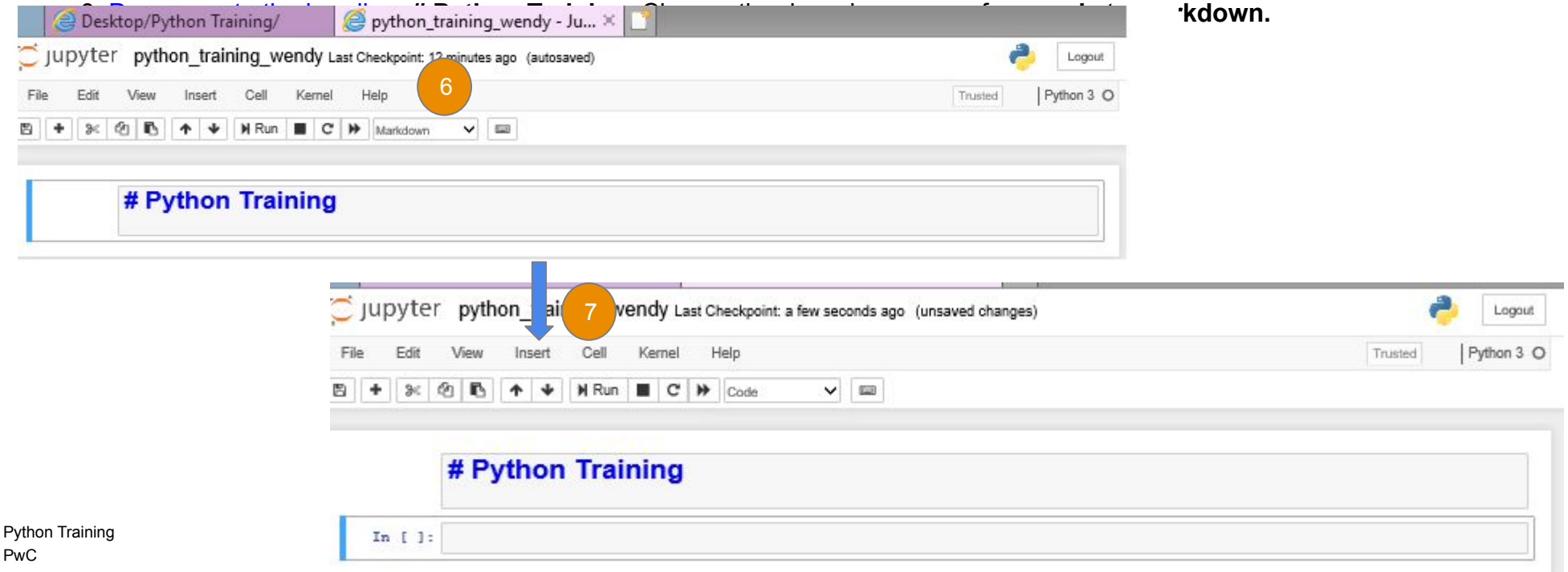
Jupyter Notebook



4. Go to File and save your file.
5. Save your file. E.g. python_training_name.ipynb

Python Style Guide for coding

Header
Header
Header
* Bullet Point



Python Style Guide for Coding

- **Python Coding Style Guide** used by Developers https://www.python.org/dev/peps/pep-0008/
- Indent code
- Add a comment
- Function
- Lists

```
# No extra indentation.
if (this_is_one_thing and
    that_is_another_thing):
    do_something()

# Add a comment, which will provide some distinction in editors
# supporting syntax highlighting.
if (this_is_one_thing and
    that_is_another_thing):
    # Since both conditions are true, we can frobnicate.
    do_something()

# Add some extra indentation on the conditional continuation line.
if (this_is_one_thing
        and that_is_another_thing):
    do_something()
```

- Function annotations should use the normal rules for colons and always have spaces around the -> arrow if present. (See Function Annotations below for more about function annotations.)

Yes:

```
def munge(input: AnyStr): ...
def munge() -> AnyStr: ...
```

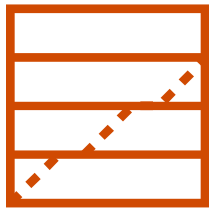
No:

```
def munge(input:AnyStr): ...
def munge()->PosInt: ...
```

1

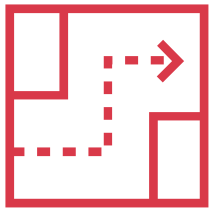
An Introduction to Python Programming

An Introduction to Python Programming



What is Python?

- Python is an open-source
- Python is free
- Python was created by Guido Van Rossum in 1991
- Object-oriented language can execute code that includes data attributes

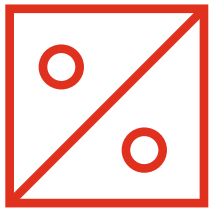


Benefits of Python

- Easy to learn coding language
- It's fast
- Kids can learn it
- Readable code that can be understood when working in large project teams
- Easy to write code for production

Why is Python popular?

- Webscraping
- Data Visualization
- Data Analysing
- Deploying apps for machine learning
- Artificial Intelligence
- Building recommendation engines



Who uses Python?

- Data Analysts
- Data Scientists
- Machine Learning Practitioners
- Software Engineers
- Roboticists
- Developers
- Researchers
- Consultants
- Programmers

2

Popular Use Cases

Popular Use Cases

Where is Python used today?	Machine Learning Algorithm
• **Recommendation Engines** e.g. Amazon, Netflix, Online shopping	Collaborative filtering
• **Virtual Assistants** e.g.Google Home, Alexa	Deep Learning (Recurrent Neural Networks) and Natural Language Processing
• **Commercial machine learning applications** e.g. customer segmentation by News Corp's readers in the cloud using AWS	K-means clustering
• **Rapid prototyping** e.g.Run an experiment which group of customers will be shown a new website by Atlassian	A/B Testing
• **Analyzing relationships** e.g. Analysing the Sydney trains network to improve Opal card services	Network Analysis, Graph databases
• **Prediction problem** e.g. Predict which class(segment) of customers Westpac will apply for a credit card	Multi-class Logistic regression
• **Data Analysis** e.g. Cleaning large datasets at PwC and reproducing the code. Reduce data entry error	Libraries Numpy (data manipulation with dataframes) and Scikit-Learn for mathematical calculations
• **Text Analytics** e.g. Allianz Insurance wants to analyze survey feedback	Natural Language Processing

Cleaning your data

Analytic Process

Analytic Process
1. Ask An Interesting Question
2. Obtain the Data
3. Understand the Data
4. Prepare the Data
5. Explore the Data
6. Model the Data
7. Evaluate the model for success

Python Data Types
• Float - real numbers
• Int - integer numbers
• Str - string, text
• Bool - True, False

Lab

Cleaning your data

1. Cleaning your data (i.e. a data frame, spreadsheet)

Import

Getting Started with pandas data frame

import module

from module import class, function, variable

```
In [2]: import numpy as np
```

Create a dataframe

```
In [ ]: df = DataFrame(data, index, columns)
```

Select a column

```
In [ ]: df[col]
```

Select row by label

```
In [ ]: df.iloc[label]
```

Delete rows or columns

```
In [ ]: df.drop()
```

View first 5 rows

```
In [ ]: df.head(5)
```

View last 5 rows

```
In [ ]: df.tail(5)
```

Sort columns

```
In [ ]: df.sort()
```

Drop rows where there is any missing data

```
In [ ]: df.dropna()
```

Count the rows for every column

```
In [ ]: df.count()
```

Cleaning the data - City of New York complaints

```
In [10]:
## Data Pre-Processing

#### We are interested in examining two columns – "Type" and "StatusDescription".
#### * Input: StatusDescription
#### * Example: "  The Department of Housing Preservation and Development inspected the following conditions. No violat
#### * Output: Type
#### * Example: NON EMERGENCY
#### Missing values are removed from "StatusDescription" column, and add a column encoding the product as an integer be

#### After cleaning up, this is the first five rows of the data we will be working on:

from io import StringIO
col = ['Type', 'StatusDescription']
df = df[col]
df = df[pd.notnull(df['StatusDescription'])]

df.columns = ['Type', 'StatusDescription']

df['TypeID'] = df['Type'].factorize()[0]
TypeID_df = df[['Type', 'TypeID']].drop_duplicates().sort_values('TypeID')

Type_to_id = dict(TypeID_df.values)
id_to_Type = dict(TypeID_df[['TypeID', 'Type']].values)

df.head()
```

Out[10]:

	Type	StatusDescription	TypeID
0	EMERGENCY	The Department of Housing Preservation and Dev...	0
1	NON EMERGENCY	The Department of Housing Preservation and Dev...	1
2	EMERGENCY	The Department of Housing Preservation and Dev...	0
3	EMERGENCY	The Department of Housing Preservation and De...	0
4	NON EMERGENCY	The Department of Housing Preservation and De...	1

4

Transforming Categorical Variables

Lab

City of New York Complaints

In [5]:

```
## Class label encoding - Transform categorical variables for Machine Learning
#### Encode class labels

from sklearn.preprocessing import LabelEncoder

l_encoder = LabelEncoder()
l_encoder.fit(y)
l_encoder.classes_
```

Out[5]:

```
array(['EMERGENCY', 'HAZARDOUS', 'IMMEDIATE EMERGENCY', 'NON EMERGENCY'],
      dtype=object)
```

One-hot encoding:

- 1 = True
- 0 = False

In [8]:

```
# Use Scikit-learn estimators for classification convert labels to integers internally
# we enumerate the class labels starting at 0:

import numpy as np

MajorCategory_mapping = {label:idx for idx,label in
                         enumerate(np.unique(df2['MajorCategory']))}
MajorCategory_mapping
{'DOOR/WINDOW':0,'UNSANITARY CONDITION':1,'PLUMBING':2,'FLOORING/STAIRS':3,'HEAT/HOT WATER':4,'PAINT/PLASTER':5, 'ELECT

Status_mapping = {label:idx for idx,label in
                  enumerate(np.unique(df2['Status']))}
Status_mapping
{'CLOSE':0,'OPEN':1}

Type_mapping = {label:idx for idx,label in
                enumerate(np.unique(df2['Type']))}
Type_mapping
{'EMERGENCY':0,'NON EMERGENCY':1, 'IMMEDIATE EMERGENCY':2, 'HARZARDOUS':3}
```

Out[8]:

```
{'EMERGENCY': 0, 'NON EMERGENCY': 1, 'IMMEDIATE EMERGENCY': 2, 'HARZARDOUS': 3}
```

Encoding other categorical variables for statistical analysis

Descriptive Statistics

Descriptive Statistics

Describe your dataframe (i.e.spreadsheet)	Definition
df.count()	Counts the row for every column
df.min()	Returns the minimum of every column
df.max()	Returns the maximum of every column
df.describe()	Generate summary statistics for every column
groupby()	Split the dataframe by columns
concat()	Merge the dataframe

Descriptive Statistics

Describe your dataframe (i.e.spreadsheet)
df.count()
df.min()
df.max()
df.describe()

Once all the categorical variables has been transformed into a vector or matrix, we can calculate summary statistics:

- Count
- Unique Count
- Min
- Max
- Mode

In [9]:

```
## Summarise the dataframe

df2.describe()
```

Out[9]:

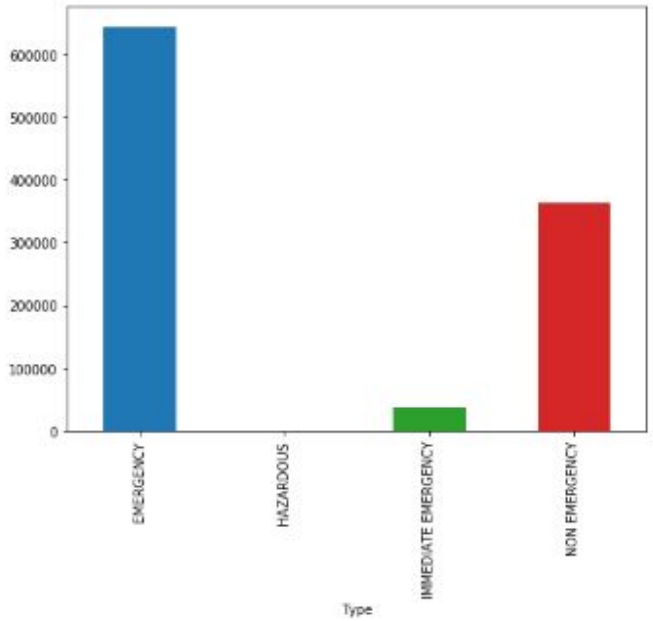
	MajorCategory	Code	Status	StatusDate	Type	MinorCategory
count	1048575	1002165	1048575	1048563	1048575	1048575
unique	16	213	2	1436	4	74
top	HEAT/HOT WATER	NO HEAT	CLOSE	1/12/15	EMERGENCY	ENTIRE BUILDING
freq	342140	204838	1046576	4855	647092	223403

Exploratory Data Analysis - Plot it and Visualise your data

In [11]:

Slide Type Slide

```
## Imbalanced Classes

#### We can see that complaint types are biased towards the complaint type 'EMERGENCY'

import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
df.groupby('Type').StatusDescription.count().plot.bar(ylim=0)
plt.show()

#### Conventional algorithms are often biased towards the majority class, not taking the data distribution into conside
```



Visualising your data - Imbalanced classes in the data!!!!

- Solution: obtain more data e.g. **synthetic dataset**

Exploratory Data Analysis - Correlations

In [13]:

Slide Type Slide

```
## Find the terms that are the most correlated with each of the products:

from sklearn.feature_selection import chi2
import numpy as np
N = 2
for Type, TypeID in sorted(Type_to_id.items()):
  features_chi2 = chi2(features, labels == TypeID)
  indices = np.argsort(features_chi2[0])
  feature_names = np.array(tfidf.get_feature_names())[indices]
  unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
  bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
  print("# '{}':".format(Type))
  print("  . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-N:])))
  print("  . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-N:])))
```

```
# 'EMERGENCY':
  . Most correlated unigrams:
. heat
. building
  . Most correlated bigrams:
. hot water
. heat hot
# 'HAZARDOUS':
  . Most correlated unigrams:
. www
. information
  . Most correlated bigrams:
. www nyc
. issued information
# 'IMMEDIATE EMERGENCY':
  . Most correlated unigrams:
. building
. violations
  . Most correlated bigrams:
. conditions violations
. violations issued
# 'NON EMERGENCY':
  . Most correlated unigrams:
. heat
. building
  . Most correlated bigrams:
. heat hot
. hot water
```

Once the encoded data is in a vector form, you may perform **word correlations** to see which words are similar.

n= 1, n= 2, n=3

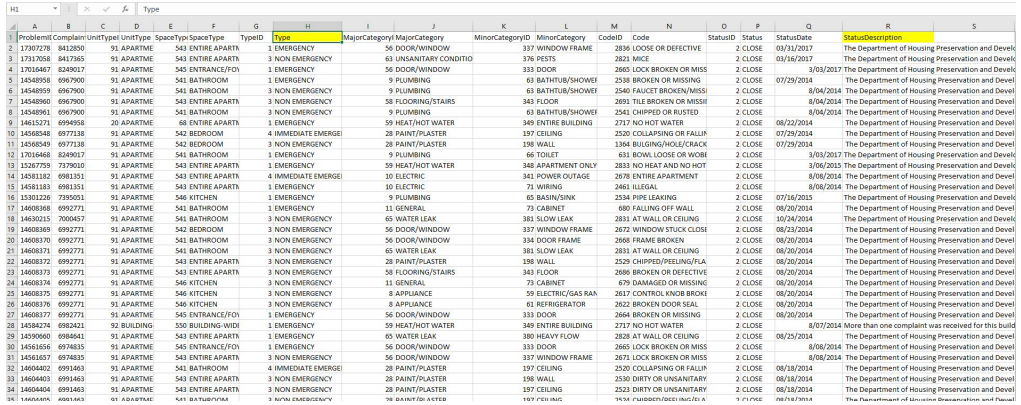
- Uni-gram
- Bi-gram
- Tri-gram

6

Text Mining

Lab

City of New York Complaints



Csv file

Dataframe

Variable

In [2]:

```
## Load the data into a Pandas Dataframe from csv file
import pandas as pd
df = pd.read_csv('Complaint_Problems.csv')
#### Inspect the first 5 values of the dataset
df.head()
```

Out[2]:

	ProblemID	ComplaintID	UnitTypeID	UnitType	SpaceTypeID	SpaceType	TypeID	Type	MajorCategoryID	MajorCategory	MinorCategoryI
0	17307278	8412850	91	APARTMENT	543	ENTIRE APARTMENT	1	EMERGENCY	56	DOOR/WINDOW	33
1	17317058	8417365	91	APARTMENT	543	ENTIRE APARTMENT	3	NON EMERGENCY	63	UNSANITARY CONDITION	37
2	17016467	8249017	91	APARTMENT	545	ENTRANCE/FOYER	1	EMERGENCY	56	DOOR/WINDOW	33
3	14548958	6967900	91	APARTMENT	541	BATHROOM	1	EMERGENCY	9	PLUMBING	6
4	14548959	6967900	91	APARTMENT	541	BATHROOM	3	NON EMERGENCY	9	PLUMBING	6

Slide Type: Slide

Data Dictionary

There were 18 variables within the data set with 1045040 rows

* ProblemID
* ComplaintID
* UnitTypeID
* UnitType
* SpaceTypeID
* SpaceType
* TypeID
* Type
* MajorCategoryID
* MajorCategory
* MinorCategoryID
* MinorCategory
* CodeID
* Code
* StatusID
* Status
* StatusDate
* StatusDescription

Tasks:

1. Insights from the dataset
* Exploratory Data Analysis

2. Build a machine learning model that performs MultiClass classification to predict the outcome of complaint type
* Type: 1=EMERGENCY 2=HARZARDOUS 3=IMMEDIATE EMERGENCY 4=NON EMERGENCY
* StatusDescription
* UnitType
* SpaceType
* MajorCategory
* MinorCategory
* Code
* Status
* StatusDate

City of New York Complaints - Transform categorical variables

In [3]:

```
## Exploratory Data Analysis

#### Create a new dataframe for exploratory data analysis

df2 = df[['MajorCategory','Code','Status','StatusDate', 'Type', 'MinorCategory']]

df2.head()
```

Out[3]:

	MajorCategory	Code	Status	StatusDate	Type	MinorCategory
0	DOOR/WINDOW	LOOSE OR DEFECTIVE	CLOSE	3/31/17	EMERGENCY	WINDOW FRAME
1	UNSANITARY CONDITION	MICE	CLOSE	3/16/17	NON EMERGENCY	PESTS
2	DOOR/WINDOW	LOCK BROKEN OR MISSING	CLOSE	3/3/17	EMERGENCY	DOOR
3	PLUMBING	BROKEN OR MISSING	CLOSE	7/29/14	EMERGENCY	BATHTUB/SHOWER
4	PLUMBING	FAUCET BROKEN/MISSING/LEAKING	CLOSE	8/4/14	NON EMERGENCY	BATHTUB/SHOWER

In [6]:

```
y_enc = l_encoder.transform(y)
np.unique(y_enc)
```

Out[6]:

```
array([0, 1, 2, 3])
```

7

Predictive Modelling

Lab

City of New York Complaints - Model Evaluation

Slide Type Slide

Model Selection

We are now ready to experiment with different machine learning models, evaluate their accuracy and find the source of any potential issues.

We will benchmark the following four models:

1. Logistic Regression

2. (Multinomial) Naive Bayes

3. Linear Support Vector Machine

4. Random Forest

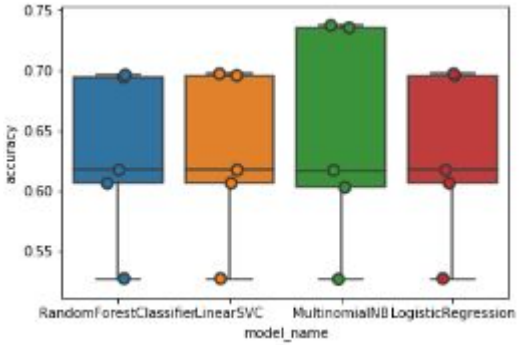
Multi-Class Model Selection

In [19]:

Slide Type Slide

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
  model_name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  for fold_idx, accuracy in enumerate(accuracies):
    entries.append((model_name, fold_idx, accuracy))
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
import seaborn as sns
sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)

plt.show()
```

- Logistic Regression
- Random Forest
- Multi-nomial Naive Bayes
- Linear Support Vector Machine



Visualise the ModelLinear Support vector

- Multinomial Naive Bayes and
- Linear Support Vector Machine

Performed better than the other classifiers

City of New York Complaints - Model Evaluation

In [22]:

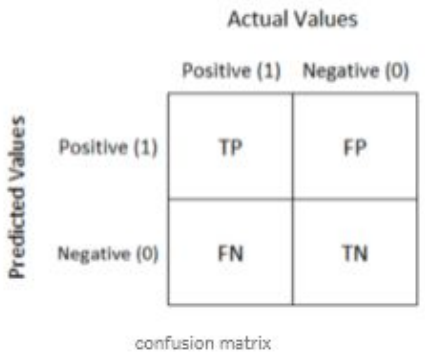
```
model = MultinomialNB()
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=TypeID_df.Type.values, yticklabels=TypeID_df.Type.values)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

In [25]:

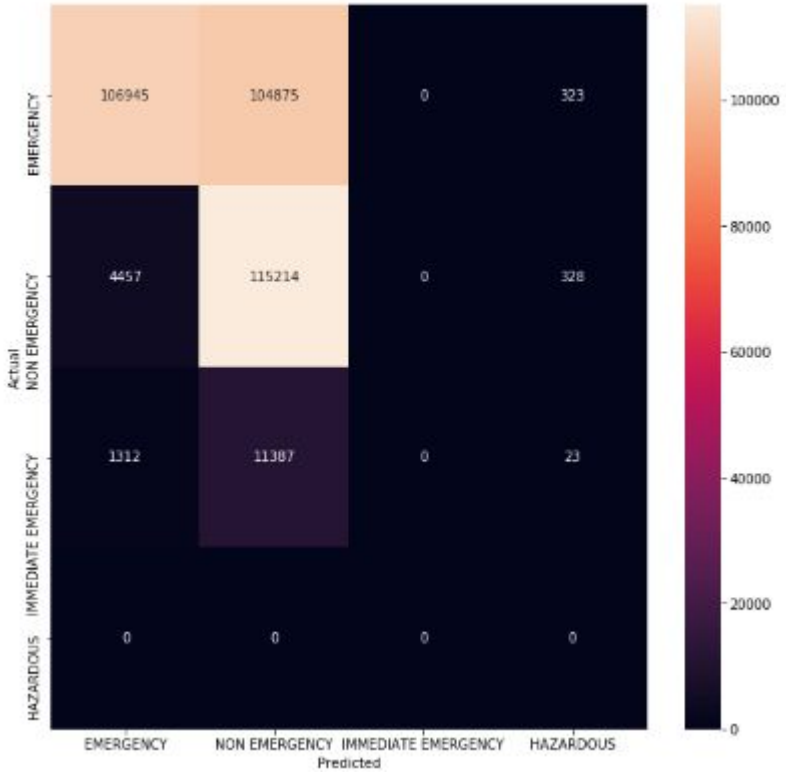
```
#### Misclassified complaint status are complaints that touch on more than one subjects (for example, complaints involv
#### Again, we use the chi-squared test to find the terms that are the most correlated with each of the categories:

model.fit(features, labels)
N = 2
for Type, TypeID in sorted(Type_to_id.items()):
  indices = np.argsort(model.coef_[TypeID])
  feature_names = np.array(tfidf.get_feature_names())[indices]
  unigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 1][:N]
  bigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 2][:N]
  print("# '{}':".format(Type))
  print("  . Top unigrams:\n       . {}".format('\n       . '.join(unigrams)))
  print("  . Top bigrams:\n       . {}".format('\n       . '.join(bigrams)))
```

```
# 'EMERGENCY':
  . Top unigrams:
       . complaint
       . violations
  . Top bigrams:
       . complaint closed
       . department housing
# 'HAZARDOUS':
  . Top unigrams:
       . violations
       . www
  . Top bigrams:
       . issued information
       . www nyc
# 'IMMEDIATE EMERGENCY':
  . Top unigrams:
       . violations
       . inspected
  . Top bigrams:
       . violations issued
       . conditions violations
# 'NON EMERGENCY':
  . Top unigrams:
       . violations
       . inspected
  . Top bigrams:
       . conditions violations
       . development inspected
```

Fitting the model



Confusion Matrix - produces **model evaluation metrics**



Top Tip - Other Model Evaluation Metrics:

- **Sensitivity**
- **Specificity (precision)**
- **F1 score**

City of New York Complaints - Naive Bayes Classifier

In [14]:

```
# Multi-Class Classifier: Features and Design

#### To train supervised classifiers, we first transformed the "StatusDesscription" into a vector of numbers. We explor
#### After having this vector representations of the text we can train supervised classifiers to train unseen "Consumer
#### After all the above data transformation, now that we have all the features and labels, it is time to train the cla

## Naive Bayes Classifier: the one most suitable for word counts is the multinomial variant:

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
X_train, X_test, y_train, y_test = train_test_split(df['StatusDescription'], df['Type'], random_state = 0)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
clf = MultinomialNB().fit(X_train_tfidf, y_train)
```

Sklearn module allows you to make a **prediction**

In [15]:

```
## After fitting the training set, let's make some predictions to predict the status of the complaint type:

print(clf.predict(count_vect.transform([" The Department of Housing Preservation and Development inspected the followin
```

```
['NON EMERGENCY']
```

Predicted Word #1

Predicted Word #2

In [16]:

```
df[df['StatusDescription'] == " The Department of Housing Preservation and Development inspected the following conditio
```

Out[16]:

	Type	StatusDescription	TypeID
8	IMMEDIATE EMERGENCY	The Department of Housing Preservation and De...	2
9	NON EMERGENCY	The Department of Housing Preservation and De...	1
12	IMMEDIATE EMERGENCY	The Department of Housing Preservation and De...	2
13	EMERGENCY	The Department of Housing Preservation and De...	0
15	EMERGENCY	The Department of Housing Preservation and De...	0
18	NON EMERGENCY	The Department of Housing Preservation and De...	1
19	NON EMERGENCY	The Department of Housing Preservation and De...	1
20	NON EMERGENCY	The Department of Housing Preservation and De...	1
21	NON EMERGENCY	The Department of Housing Preservation and De...	1
23	NON EMERGENCY	The Department of Housing Preservation and De...	1
24	NON EMERGENCY	The Department of Housing Preservation and De...	1
30	IMMEDIATE EMERGENCY	The Department of Housing Preservation and De...	2
31	NON EMERGENCY	The Department of Housing Preservation and De...	1
32	NON EMERGENCY	The Department of Housing Preservation and De...	1
33	NON EMERGENCY	The Department of Housing Preservation and De...	1
34	NON EMERGENCY	The Department of Housing Preservation and De...	1
35	EMERGENCY	The Department of Housing Preservation and De...	0
36	EMERGENCY	The Department of Housing Preservation and De...	0
37	EMERGENCY	The Department of Housing Preservation and De...	0
38	EMERGENCY	The Department of Housing Preservation and De...	0
39	EMERGENCY	The Department of Housing Preservation and De...	0
42	NON EMERGENCY	The Department of Housing Preservation and De...	1
44	NON EMERGENCY	The Department of Housing Preservation and De...	1
48	EMERGENCY	The Department of Housing Preservation and De...	0
53	EMERGENCY	The Department of Housing Preservation and De...	0
54	EMERGENCY	The Department of Housing Preservation and De...	0
59	IMMEDIATE EMERGENCY	The Department of Housing Preservation and De...	2
71	EMERGENCY	The Department of Housing Preservation and De...	0
72	EMERGENCY	The Department of Housing Preservation and De...	0
77	EMERGENCY	The Department of Housing Preservation and De...	0
...	...	...	...
53280	NON EMERGENCY	The Department of Housing Preservation and De...	1
53281	NON EMERGENCY	The Department of Housing Preservation and De...	1

Data Visualization

Lab

City of New York Complaints - Data Visualization

In [29]:

Slide Type: Slide

```
import seaborn as sns
sns.set(color_codes=True)
```

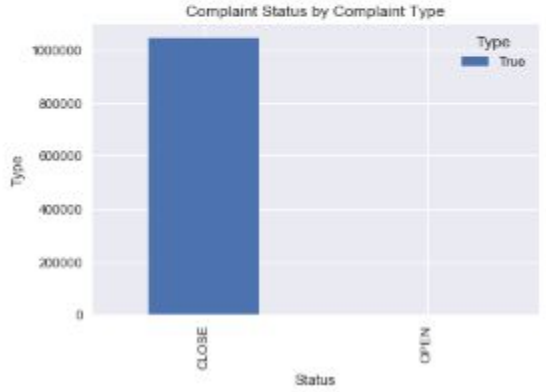
Import module for data visualization **seaborn**

In [43]:

Slide Type: Slide

```
# Barplot of Status grouped by Type
pd.crosstab(df2.Status, df2.Type.astype(bool)).plot(kind='bar')
plt.title('Complaint Status by Complaint Type')
plt.xlabel('Status')
plt.ylabel('Type')

# status -> True = Open, False = Closed
```

Out[43]: `Text(0,0.5,'Type')`



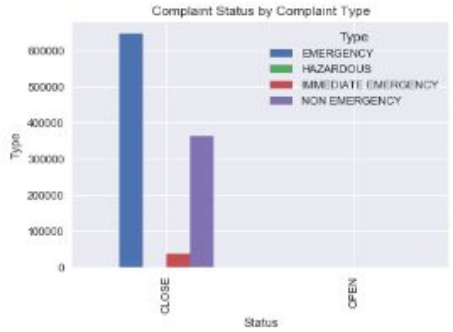
Bar plot

City of New York Complaints - Data Visualization

In [44]:

```
# Barplot of Status grouped by Type
pd.crosstab(df2.Status, df2.Type).plot(kind='bar')
plt.title('Complaint Status by Complaint Type')
plt.xlabel('Status')
plt.ylabel('Type')
```

Out[44]: Text(0,0.5,'Type')



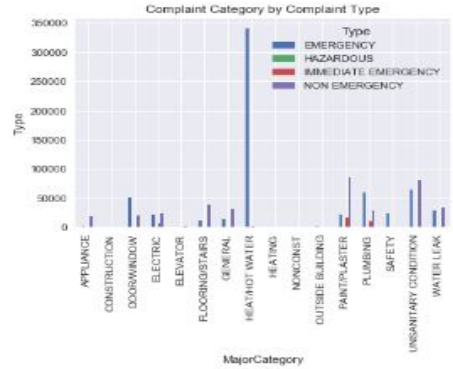
Bar plot - grouped data

Bar plot - Most of the complaints relate to **hot water**

In [38]:

```
# Barplot of MajorCategory grouped by Type
pd.crosstab(df2.MajorCategory, df2.Type).plot(kind='bar')
plt.title('Complaint Category by Complaint Type')
plt.xlabel('MajorCategory')
plt.ylabel('Type')
```

Out[38]: Text(0,0.5,'Type')



9

Turning your code into a client presentation

Lab

Turn your Jupyter Notebook into slides

- **File -> Download as -> Slides (slides.html)**

In Windows

- **A short tutorial for slide conversion in Mac:**
- https://medium.com/learning-machine-learning/present-your-data-science-projects-with-jupyter-slides-75f20735eb0f

In Mac

10

Equivalent code in R programming

Equivalent Code in R Programming

Lab - Next Time

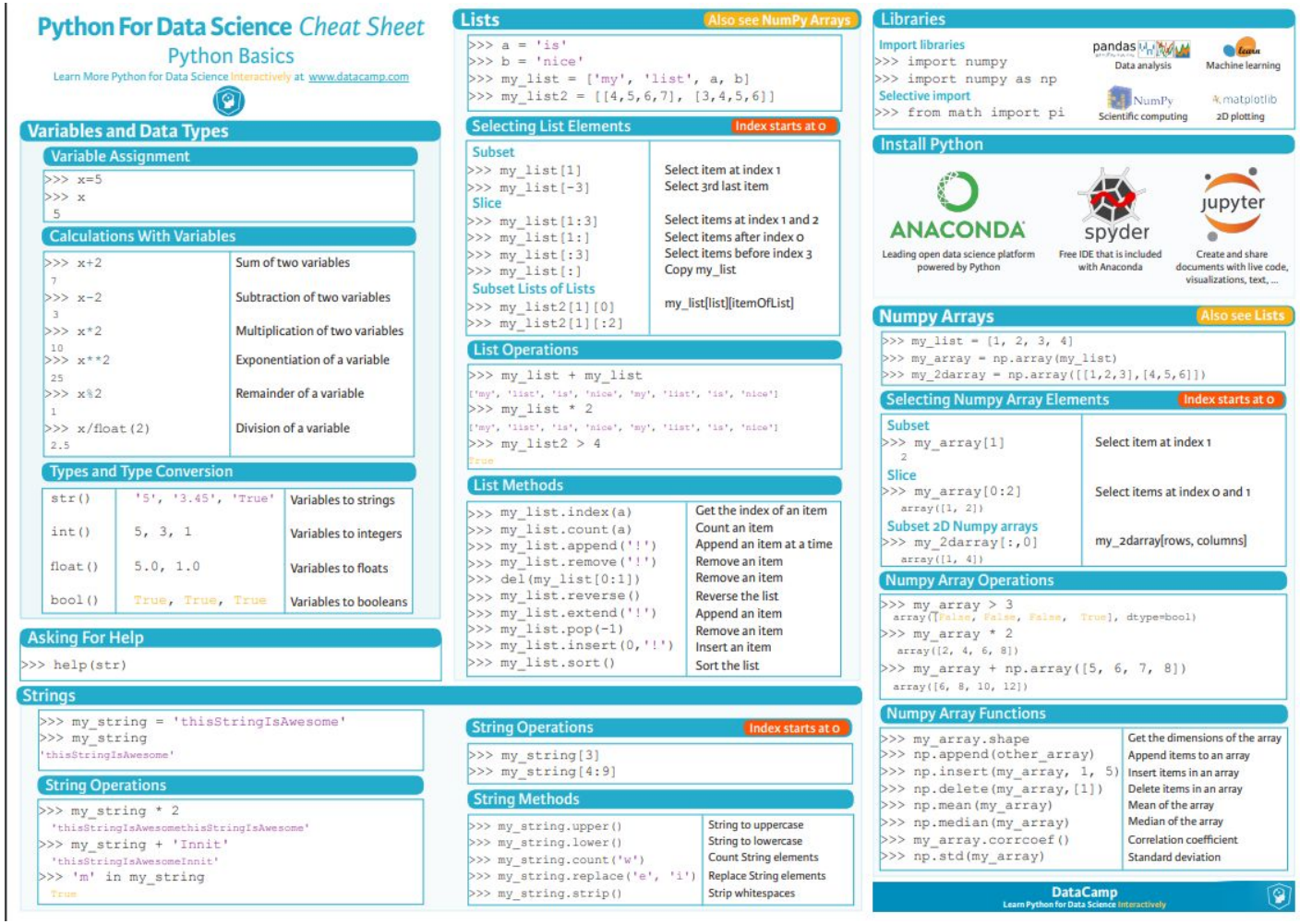
11

Bonus

Bonus

Additional Resources for Learning	From Beginners to Advanced Learning
Online courses	• **Introduction to Python Programming** FREE https://www.udemy.com/course/pythonforbeginnersintro/
	• Advanced course - **Python 3 Complete Masterclass - Make Your Job Tasks Easier! $21.99** https://www.udemy.com/course/pythontutorial/
	• KhanAcademy.com, Codeacademy.com, PwC Vantage, Coursera, LinkedIn Learning and EdX
Face to Face courses	• Data Science at General Assembly Sydney
Books	• Python for Data Analysis by Wes McKinney
	• Python for Machine Learning by Sebastian Raschka and Vahid Mirjalili
Cheat Sheet	• Python for Data Science cheat sheet https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf
Kaggle competition	• Learn via Kaggle competitions https://www.kaggle.com/competitions

Python Cheat Sheet - Data Camp



Python For Data Science *Cheat Sheet*

Python Basics

Learn More Python for Data Science Interactively at www.datacamp.com

Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
5
```

Calculations With Variables

```
>>> x+2
7
>>> x-2
3
>>> x*2
10
>>> x**2
25
>>> x%2
1
>>> x/float(2)
2.5
```

Sum of two variables
Subtraction of two variables
Multiplication of two variables
Exponentiation of a variable
Remainder of a variable
Division of a variable

Types and Type Conversion

`str()`	`'5', '3.45', 'True'`	Variables to strings
`int()`	`5, 3, 1`	Variables to integers
`float()`	`5.0, 1.0`	Variables to floats
`bool()`	`True, True, True`	Variables to booleans

Asking For Help

```
>>> help(str)
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset

```
>>> my_list[1]
>>> my_list[-3]
```

Select item at index 1
Select 3rd last item

Slice

```
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
```

my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

`>>> my_list.index(a)`	Get the index of an item
`>>> my_list.count(a)`	Count an item
`>>> my_list.append('!')`	Append an item at a time
`>>> my_list.remove('!')`	Remove an item
`>>> del(my_list[0:1])`	Remove an item
`>>> my_list.reverse()`	Reverse the list
`>>> my_list.extend('!')`	Append an item
`>>> my_list.pop(-1)`	Remove an item
`>>> my_list.insert(0,'!')`	Insert an item
`>>> my_list.sort()`	Sort the list

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

`>>> my_string.upper()`	String to uppercase
`>>> my_string.lower()`	String to lowercase
`>>> my_string.count('w')`	Count String elements
`>>> my_string.replace('e', 'i')`	Replace String elements
`>>> my_string.strip()`	Strip whitespaces

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

Selective import

```
>>> from math import pi
```

pandas — Data analysis
scikit-learn — Machine learning
NumPy — Scientific computing
matplotlib — 2D plotting

Install Python

ANACONDA — Leading open data science platform powered by Python
spyder — Free IDE that is included with Anaconda
jupyter — Create and share documents with live code, visualizations, text, ...

Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset

```
>>> my_array[1]
2
```

Select item at index 1

Slice

```
>>> my_array[0:2]
array([1, 2])
```

Select items at index 0 and 1

Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
array([1, 4])
```

my_2darray[rows, columns]

Numpy Array Operations

```
>>> my_array > 3
array([False, False, False,  True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

Numpy Array Functions

`>>> my_array.shape`	Get the dimensions of the array
`>>> np.append(other_array)`	Append items to an array
`>>> np.insert(my_array, 1, 5)`	Insert items in an array
`>>> np.delete(my_array,[1])`	Delete items in an array
`>>> np.mean(my_array)`	Mean of the array
`>>> np.median(my_array)`	Median of the array
`>>> my_array.corrcoef()`	Correlation coefficient
`>>> np.std(my_array)`	Standard deviation

DataCamp
Learn Python for Data Science Interactively

References

In [ ]:

Slide Type: Slide

```
## References:
* https://scikit-learn.org/stable/modules/multiclass.html
* https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f
* https://towardsdatascience.com/beginners-guide-to-lda-topic-modelling-with-r-e57a5a8e7a25
* https://medium.com/learning-machine-learning/present-your-data-science-projects-with-jupyter-slides-75f20735eb0f
```

Thank you

pwc.com

© 2018 PwC. All rights reserved. Not for further distribution without the permission of PwC. “PwC” refers to the network of member firms of PricewaterhouseCoopers International Limited (PwCIL), or, as the context requires, individual member firms of the PwC network. Each member firm is a separate legal entity and does not act as agent of PwCIL or any other member firm. PwCIL does not provide any services to clients. PwCIL is not responsible or liable for the acts or omissions of any of its member firms nor can it control the exercise of their professional judgment or bind them in any way. No member firm is responsible or liable for the acts or omissions of any other member firm nor can it control the exercise of another member firm’s professional judgment or bind another member firm or PwCIL in any way.