Python Cheatsheet

Samntha Staheli

Contents

Import Libraries

Basic most used:

```
import pandas as pd
import altair as alt
import numpy as np
```

Data:

```
import datadotworld as dw
```

For Machine Learning Models:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

JSON:

```
alt.data_transformers.enable('json')
```

Data from URL

We load data by using functions such as read_csv() and read_json().

csv example:

```
url = 'https://url-data.com'
data = pd.read_csv(url)
```

JSON example:

```
url = 'https://url-data.com'
data = pd.read_json(url)
```

Markdown Formatting

Headers

Headers are made by putting a # before the text. The amount of # before determines the size. Examples are found bellow.

Header 1

Header 2

Header 3

Header 4

Images

Load images by typeing . Make sure to use correct name and include .png or .jpg.

example:

```
![](christmas-tree.jpg)
```

This code would load this image:

Italics

Italics are made by putting the text inbetween underscores.

italic text

Bold

Bold text is similar to Italics but use two underscores instead.

bold text

Lists

There are three ways to create list.

First way:

```
- item 1
- item 2
- item 3
```

Result:

- item 1
- item 2
- item 3

Second way:

```
* item 1
* item 2
* item 3
```

Result:

- item 1
- item 2
- item 3

Third way:

```
1. item 1
2. item 2
3. item 3
```

Result:

1. item 1
2. item 2
3. item 3

Code Snippets

Code snippets are made by putting the code between three backtics. You can also include the programming language by typeing the name after the first three backtics.

Tables

example:

```
| airport_code   |   total_flights |   total_delays |   total_delay_min |   prop_delays |   avg_delay_hrs |
|:---------------|----------------:|---------------:|------------------:|--------------:|----------------:|
| ATL            |         4430047 |         902443 |          53983926 |      0.20371  |        0.996996 |
| DEN            |         2513974 |         468519 |          25173381 |      0.186366 |        0.895495 |
| IAD            |          851571 |         168467 |          10283478 |      0.197831 |        1.01736  |
| ORD            |         3597588 |         830825 |          56356129 |      0.230939 |        1.13053  |
| SAN            |          917862 |         175132 |           8276248 |      0.190804 |        0.78762  |
| SFO            |         1630945 |         425604 |          26550493 |      0.260955 |        1.03972  |
| SLC            |         1403384 |         205160 |          10123371 |      0.146189 |        0.822396 |
```

result:

airport_code	total_flights	total_delays	total_delay_min	prop_delays	avg_delay_hrs
ATL	4430047	902443	53983926	0.20371	0.996996
DEN	2513974	468519	25173381	0.186366	0.895495
IAD	851571	168467	10283478	0.197831	1.01736
ORD	3597588	830825	56356129	0.230939	1.13053
SAN	917862	175132	8276248	0.190804	0.78762
SFO	1630945	425604	26550493	0.260955	1.03972
SLC	1403384	205160	10123371	0.146189	0.822396

Altair charts

Tables

example of basic format:

```
table = (data
    .groupby('a column name')
    .agg(total = ('total', sum),
        mean = ('mean', mean),
        max = ('max', max),
        min = ('min', min)
        )
    .assign(another_column_can_make = lambda x: function goes here)
    .reset_index()
)
```

make table markdown format:

```
print(table.to_markdown())
```

make table markdown format without index numbers:

```
print(table.to_markdown(index=False))
```

Charts

```
chart = (alt.Chart(data)
    .mark_area(opacity=number between 0-1, color='color name')
    .encode(x = alt.X('year', axis = alt.Axis(format = 'd', title = 'Year')), y = 'Total')
    .properties(
        height = 150,
        width = 500,
        title = {'text': 'Chart Title', 'subtitle': 'chart subsitle'}
    )
)
```

Machine Learning Model

To create a machine learning model we use the sklearn library. The types you can use are:

- DecisionTreeClassifier
- RandomForestClassifier
- GradientBoostingClassifier

The following code is from the model made in project 5.

First you craete features and targets.

```
features = dwellings_ml.drop(columns=['before1980', 'yrbuilt', 'parcel'])
targets = dwellings_ml.before1980
```

Create test and train valriables.

x = features

y = targets

Random state is like setting seed in R.

```
x_train, x_test, y_train, y_test = train_test_split(
    features,
    targets,
    test_size = .3,
    random_state = 24601)
```

Create Desison Tree.

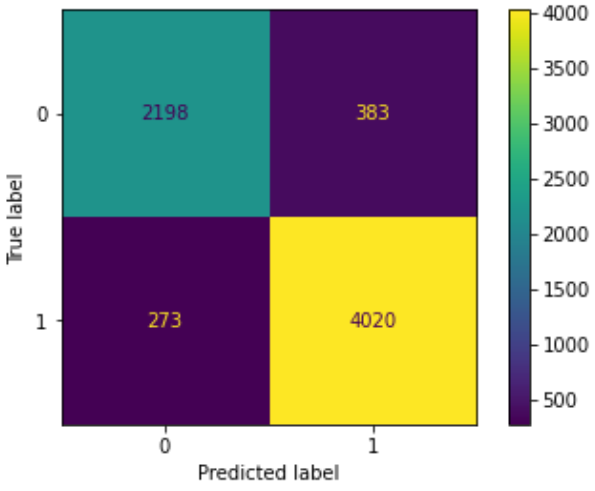
This example uses the gradient boosting classifer. The result is a number between 0 and 1. The closer the number is to 1 the more accurate the model was.

```
# create a classification model
classifier_GB = GradientBoostingClassifier()

# train the model
classifier_GB.fit(x_train, y_train)

# use your model to make predictions!
y_predicted = classifier_GB.predict(x_test)

# test how accurate those predictions are
metrics.accuracy_score(y_test, y_predicted)
```

Create Model Matrix.

```
metrics.plot_confusion_matrix(classifier_GB, x_test, y_test)
```

result:



Calculate precision, recall, f1-score, and balance accuracy.

```
print(metrics.classification_report(y_test, y_predicted))
```

result:
precision recall f1-score support

```
        0       0.89      0.85      0.87      2581
        1       0.91      0.94      0.92      4293

accuracy                            0.90      6874
```

macro avg 0.90 0.89 0.90 6874
weighted avg 0.90 0.90 0.90 6874

SQL Queries

To create SQL queries in python we use the dataworld library.

example from project 3:

```
byui_players = dw.query('byuidss/cse-250-baseball-database',
'''
SELECT DISTINCT salaries.playerid, salaries.salary, salaries.teamid, salaries.yearid, collegeplaying.
FROM salaries AS sal
    JOIN collegeplaying AS cp ON sal.playerid = cp.playerid
 WHERE collegeplaying.schoolid = "idbyuid"
 ORDER BY salaries.salary
''')
q1_table = byui_players.dataframe
```

SQL Basics

Difference between SELECT and SELECT DISTINCT:
SELECT DISTINCT gets rid of duplacites, while SELECT returns all results.

Difference between WHERE and HAVING:
WHERE uses the selected data, while HAVING uses data created in select statment.

Difference between GROUP BY and ORDER BY:
GROUP BY totals results based on column stated, while ORDER BY orders the results based on the column stated. ORDER BY's default is ASC but can change to DESC.