PXL-Digital

Streamlit Applicaties

Streamlit applicaties runnen in notebook en deployen op Heroku

Christiaan Prévot, Dennis Keusters, Kristof Heulsen
6-1-2020

Doelstelling:

De bedoeling van deze opdracht is om verschillende notebook files aan te maken, die samenwerken met verschillende files (csv, pickle, images), Als alle files werken gaan we deze deployen op Heroku zodat ze online staan en je ze altijd kan raadplegen. Eens de apllicaties online staan kan je bekijken en er interactief mee kan werken.

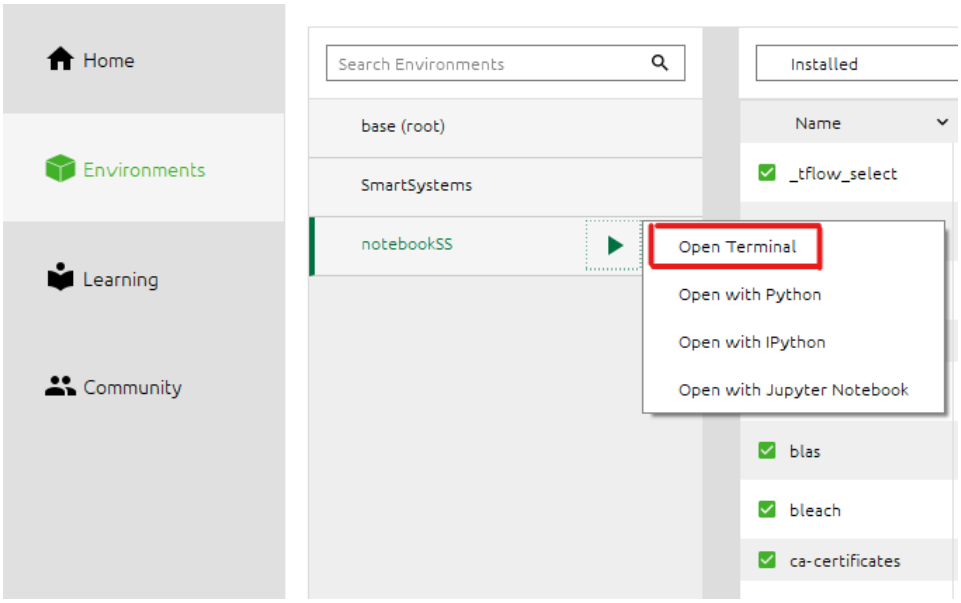
Verder beschrijft deze handleiding hoe men streamlit applicaties kan openen En hoe je zelf een eigen streamlit applicatie via notebook op heroku kan deployen. Ook word er uitgelegd hoe de zelfgemaakte streamlit applicaties werken en wat ze doen.

De code kan gedownload worden via de volgende link : https://github.com/ultratronics/smart_systems_eindoefening?fbclid=IwAR2Y_BTy7itvyAxPCPPQBdXOBY4JYhO43q9HgJHpkZ5GafjNo_9Ic2lcQvY

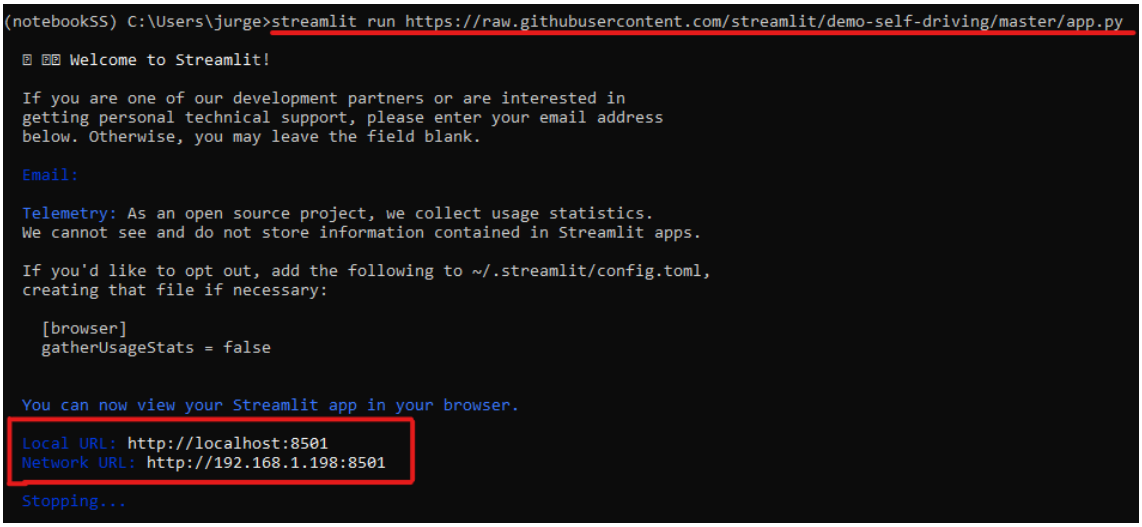
Demo applicaties:

Openen van streamlit applicatie demo Self-driving Car

1. Open terminal.

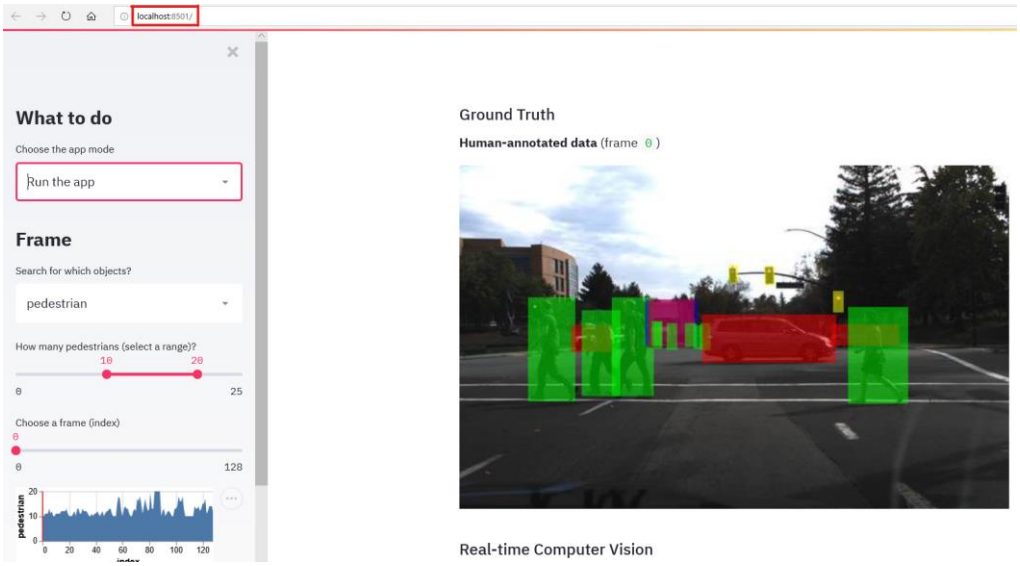


2. Pip install Streamlit.
3. pip install --upgrade streamlit opencv-python.
4. streamlit run https://raw.githubusercontent.com/streamlit/demo-self-driving/master/app.py



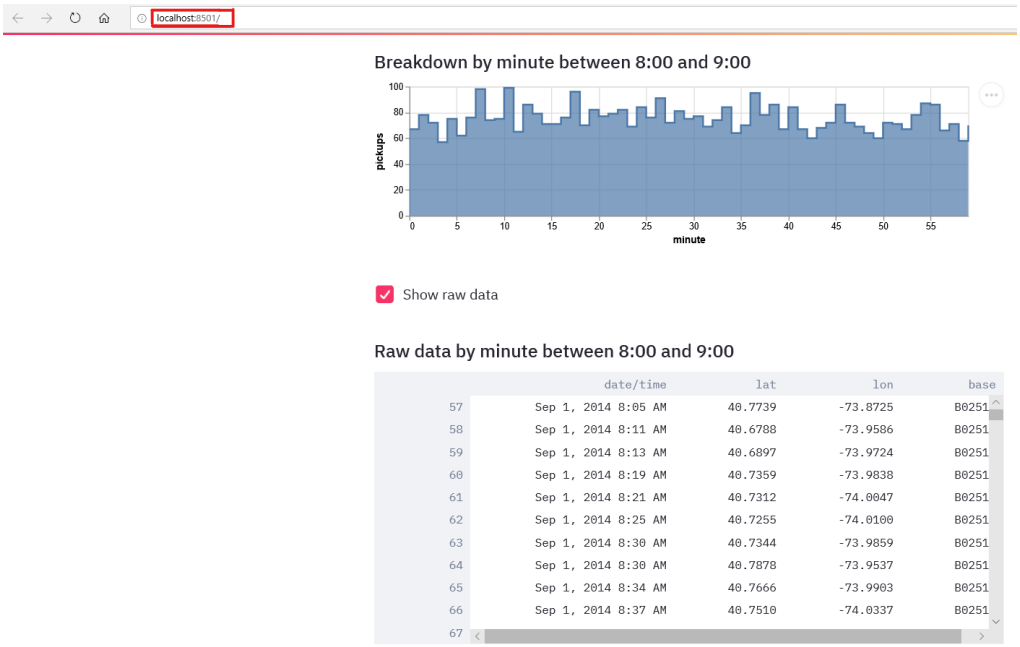
5. Normaal start de applicatie vanzelf anders vind je deze in de opgegeven ip-adres.

6. De applicatie gaat open in een internet tablad.
7. Links zie je de data inputs, dit zijn verschillende selectie-/ inputboxen en sliders die je kan veranderen om op de foto rechts andere dingen weer te geven. Dit zorgt voor een directe aanpassing die zichtbaar is op je app.



Openen van streamlit applicatie demo uber drives

1. Open terminal (zie vorige).
2. pip install --upgrade streamlit (indien er een nieuwe versie is, anders ga je naar stap 3).
3. streamlit run https://raw.githubusercontent.com/streamlit/demo-uber-nyc-pickups/master/app.py



Deploy een streamlit applicatie op Heroku

1. Open een jupyter notebook.
2. Importeren van de juiste libraries.

```
import streamlit as st
import plotly as px
from PIL import Image
#import cv2
#print (cv2.__version__)
#st.text('hello test 123 does this work?')

# Ignore warnings :
#import warnings
#warnings.filterwarnings('ignore')


# Handle table-like data and matrices :
import numpy as np
import pandas as pd
import math
```

3. Inlezen van een pickle in notebook.

```
clf_rf = RandomForestRegressor()
#clf_rf.fit(X_train , y_train)
import pickle
clf_rf = pickle.load(open("Model.pkl", "+rb"))
```

4. Converteren de notebook file naar een python file en sla het op een gekende plaats op.

```
!jupyter nbconvert --output-dir='.\push' --to script diamonds-in-depth-analysis.ipynb
```

```
[NbConvertApp] Converting notebook diamonds-in-depth-analysis.ipynb to script
[NbConvertApp] Writing 21810 bytes to .\push\diamonds-in-depth-analysis.py
```

5. Snij de laatste commando's weg uit de file.
    a. Linux commando's: Deze werken dus niet standaard op windows. Vervolgens maken we een copy van deze file en slaan hem op in een andere file die dan naar Heroku wordt gepushd.

```
sed '757,759d' .\push\diamonds-in-depth-analysis.py
tailer.head( -n -9 .\push\diamonds-in-depth-analysis.py > .\push\pushreal\diamonds-in-depth-analysis2.py )
tailer.head(open() -n -9 .\push\diamonds-in-depth-analysis.py > .\push\pushreal\diamonds-in-depth-analysis2.py )
```

    b. Python commando's: Deze werken zowel in windows als in linux.

```
readFile = open(".\push\pushreal\diamonds-in-depth-analysis2.py")

lines = readFile.readlines()

readFile.close()
w = open(".\push\pushreal\diamonds-in-depth-analysis2.py",'w')
w.writelines([item for item in lines[:-40]])
w.close()
```

6. Push nu de hele map naar heroku via git commit en git push (je kunt dit best twee maal achter elkaar doen want soms vindt hij bij de eerste push geen veranderingen).

```
!git -C ./push/pushreal commit -am "Nieuw commit" | git -C ./push/pushreal push heroku master
```

```
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Python app detected
remote: -----> Installing SQLite3
remote: Sqlite3 successfully installed.
remote: -----> Installing requirements with pip
remote:
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 166.3M
remote: -----> Launching...
remote:        Released v45
remote:        https://eindoefening.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/eindoefening.git
   e9bc234..8a2a24a  master -> master
```

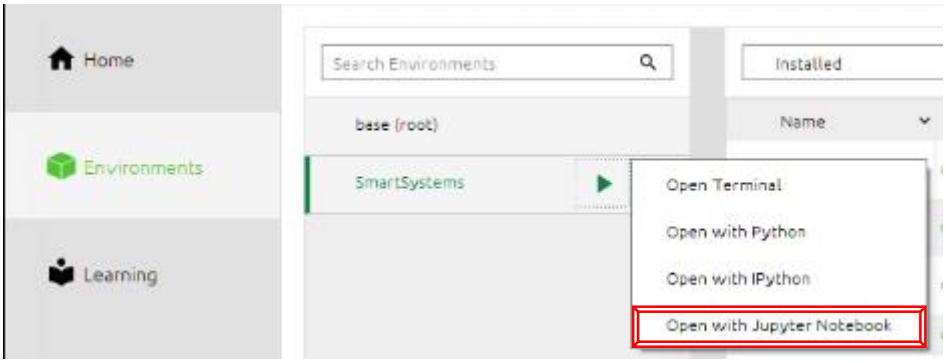
```
!git -C ./push/pushreal commit -am "Nieuw commit" | git -C ./push/pushreal push heroku master
```

Uitleg eigen applicaties:

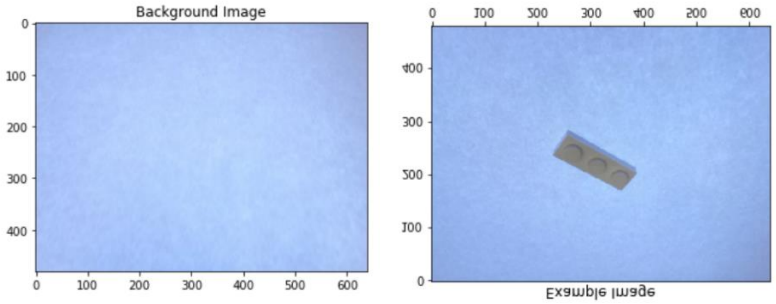
Lego Dataset

Het doel van deze notebook is om verschillende soorten lego blokken te herkennen aan de hand van verschillende foto's. Dit gaan we doen door middel van OpenCV.

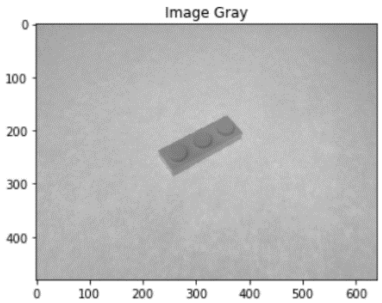
1. Open anaconda
2. Open uw environment
3. Open met jupyter notebook



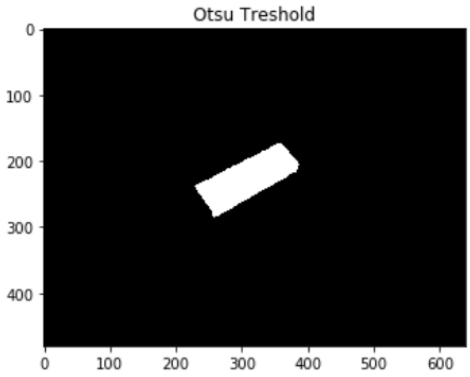
4. De notebook gaat open in een internet tablad.
5. Selecteer de map en open de python file.
6. Run de code met ▶▶ .
7. Als de applicatie klaar is.
    a. Zie je afbeeldingen van bepaalde lego blokken en van de achtergrond.



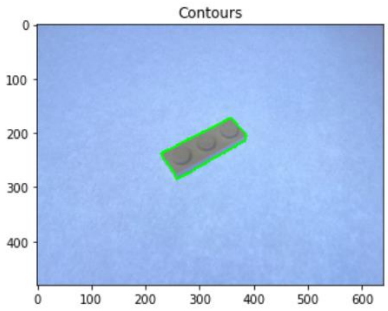
    b. Vervolgens worden de afbeeldingen in Grayscale gezet.

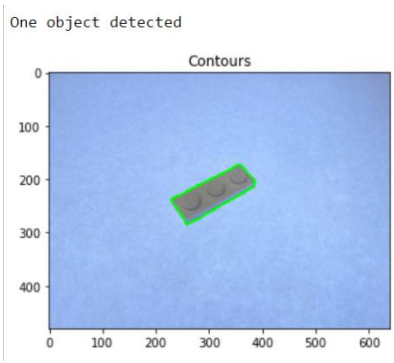


    c. De achtergrond word zwart en het object zelf word wit gemaakt zodat het onderscheid tussen object en achtergrond groot is.



d. Er word een contour gemaakt rond het object dit is handig als er meerdere objecten zijn om ze zo gemakkelijk uit elkaar te halen en om te weten hoeveel objecten er zijn.

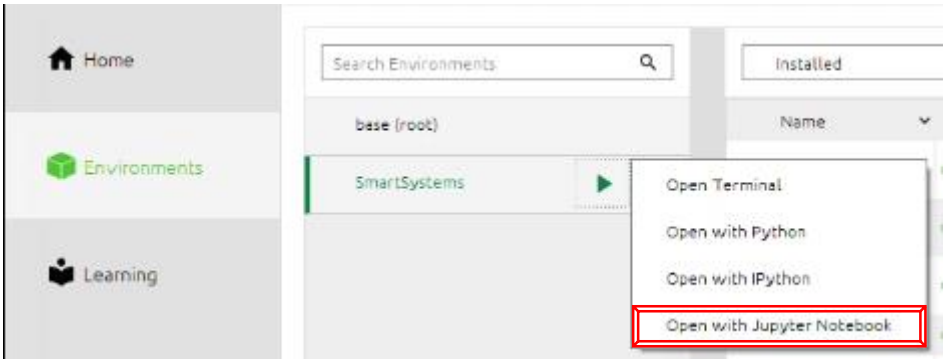




Automatische Pickle file

In deze applicatie gaan we verschillende csv en pickle files online zetten op heroku, zodat je deze kan gaan gebruiken als streamlit applicatie.

1. Open anaconda
2. Open uw environment
3. Open met jupyter notebook



4. De notebook gaat open in een internet tablad.
5. Selecteer de map en open de python file.
6. Run de code met ⏩ .
    a. Er worden files ingelezen (csv en pickle)

```
# lees de pckle en csv file in.
model =pickle.load(open("./BreastCancerPickle.pkl","+rb"))
test = pd.read_csv("./wisc_bc_data(1).csv")
st.write(test.dtypes)
```

    b. Een functie controleert of er geen lege string gebruikt wordt.

```
def checkIfStringEmpty(var):
    if var == "":
        return 0
    else:
        return var
```

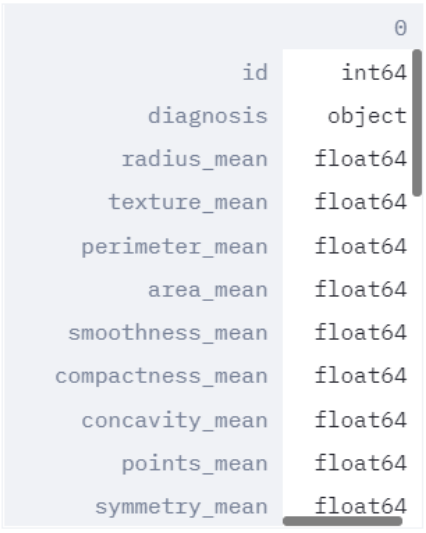
    c. We doorlopen alle kolommen die er zijn in de csv file.

```
aantal_kolommen = len(test.columns)
output_categorie = "diagnosis"
index_output = test.columns.get_loc(output_categorie)
inputlist = []
list1 = []  # np.empty((0,aantal_kolommen-1))
outputnaam = ""

# toekennen van de ingangen en de uitgangen
for i in range(0, aantal_kolommen):
    # print(i)
    # print(index_output)
    if i == index_output:
        list1.append("output")
    else:
        list1.append("input")
print(list1)

st.write("hallo")
```

    d. We kennen aan elke kolom toe of het een ingang of een uitgang is.
    e. Daarna converteren we de notebook file naar een python file.
        i. We verwijderen de laatste regels omdat deze na het runnen niet meer uitgevoerd mogen worden.
    f. We pushen de python file daarna naar heroku, waardoor er een streamlit applicatie aangemaakt wordt die je kan raadplegen.

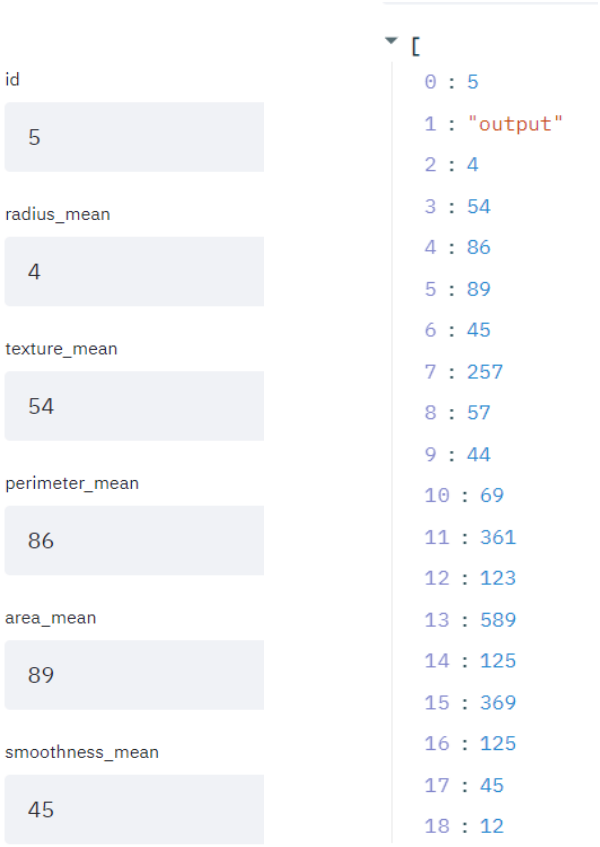
```
!jupyter nbconvert --output-dir='./push' --to script autopickl.ipynb
!head -n -7 ./push/autopickl.py > ../automatisch_pickle/project.py
!git -C ../automatisch_pickle commit -am "Nieuw commit" | git -C ../automatisch_pickle push heroku master
!git -C ../automatisch_pickle commit -am "Nieuw commit" | git -C ../automatisch_pickle push heroku master
```

7. Als de applicatie klaar is kan je applicatie openen.
    a. Je kan de kolommen zien en welke type elke kolom is.



	0
id	int64
diagnosis	object
radius_mean	float64
texture_mean	float64
perimeter_mean	float64
area_mean	float64
smoothness_mean	float64
compactness_mean	float64
concavity_mean	float64
points_mean	float64
symmetry_mean	float64

    b. Je kan alle inputs een waarde meegeven en zo een eigen output waarde berekenen.
    Input:



Output:

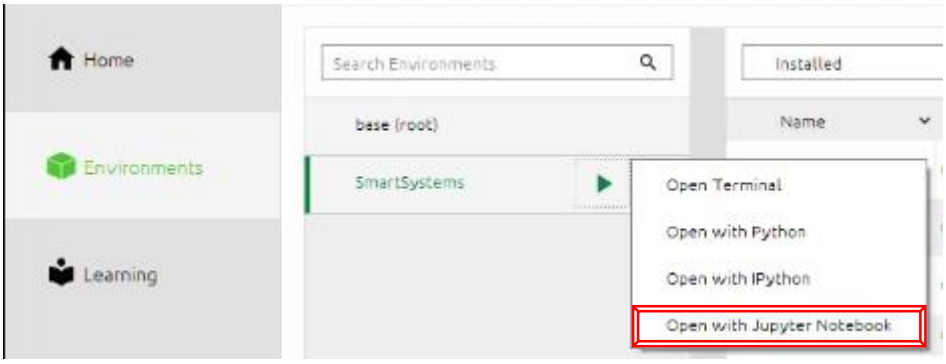
```
<class 'list'>
```

	0
0	5
2	4.0
3	54.0
4	86.0
5	89.0
6	45.0
7	257.0
8	57.0
9	44.0
10	69.0
11	361.0

Diamond

In deze applicatie gaan we een notebook maken waarin we bepaalde details van diamanten gaan bereken, ook kan deze data visueel worden weergegeven op grafieken en heatmaps.

1. Open anaconda
2. Open uw environment
3. Open met jupyter notebook



4. De notebook gaat open in een internet tablad.
5. Selecteer de map en open de python file.
6. Run de code met ▶▶ .
7. De juiste libraries en csv file worden ingelezen.
8. We lezen de eerste 5 rijen in om te kijken wat er allemaal in de csv file zit.

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

9. Daarna deleten we de Unnamed:0 kolom omdat we al een index kolom in het begin hebben.
10. Dan laten we verschillende grafieken zien in de notebook als visualisatie
11. De file wordt geconverteerd naar een python file.

```
!jupyter nbconvert --output-dir='.\push' --to script diamonds-in-depth-analysis.ipynb
```

```
[NbConvertApp] Converting notebook diamonds-in-depth-analysis.ipynb to script
[NbConvertApp] Writing 22228 bytes to .\push\diamonds-in-depth-analysis.py
```

12. Verwijderen we de laatste regels van de file omdat hierin opnieuw de convert en de push naar heroku staan en dat moet hier niet meer.
    a) Linux:
    De file word bewerkt en automatich naar de nieuwe locatie gezet.

```
# Linux
#sed '757,759d' .\push\diamonds-in-depth-analysis.py
tailer.head( -n -9 .\push\diamonds-in-depth-analysis.py > .\push\pushreal\diamonds-in-depth-analysis2.py )
#tailer.head(open() -n -9 .\push\diamonds-in-depth-analysis.py > .\push\pushreal\diamonds-in-depth-analysis2.py )
```

    b) Python:
    Enkel de laatste lijnen worden verwijdert.

```
readFile = open(".\push\pushreal\diamonds-in-depth-analysis2.py")

lines = readFile.readlines()

readFile.close()
w = open(".\push\pushreal\diamonds-in-depth-analysis2.py",'w')
w.writelines([item for item in lines[:-40]])
w.close()
```

    c) Als je geen van beide doet moet je file zelf kopiëren naar de juiste locatie en vervolgens bewerken om de laatste regels die Ipython() beat te verwijderen.
13. Als laatste wordt de python file gepusht naar Heroku om zo de streamlit applicatie te runnen. Deze regel moet 2 keer uitgevoerd worden omdat hij soms bij de eerste push niet altijd de veranderingen waarneemt.

```
!git -C ./push/pushreal commit -am "Nieuw commit" | git -C ./push/pushreal push heroku master
```

De streamlit code in de notebook

1) Eerst geven we een titel mee voor onze pagina.

```
# Title van Diamenten
st.title('Data Analyse van Diamonds met Streamlit')
```

2) Maken een selectbox met alle kolom inputs om zo de data te tonen

```
# display data
Keuze = { "info": df.info(),
        "head": df.head(),
        "tail": df.tail(),
        "describe": df.describe(),
        "isnull": df.isnull().sum(),
        "loc": df.loc[(df['x']==0) | (df['y']==0) | (df['z']==0)] }

Selected = st.sidebar.selectbox("Kies Wat je wilt zien", list(Keuze.keys()),0)
st.write(Keuze[Selected], use_column_width= True, caption= Keuze[Selected])
```

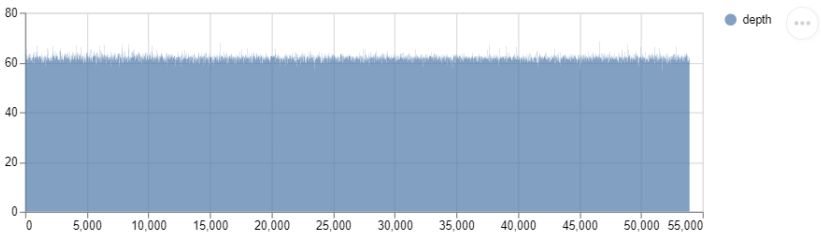
3) Als je standaard grafieken van de notebook file wilt displayen op de applicatie moet je er gewoon “st.pyplot() ” achter zetten. Dit werkt voor elke grafiek en heatmap.

```
# Correlation Map
corr = df.corr()
sns.heatmap(data=corr, square=True , annot=True, cbar=True)
st.pyplot()
```

4) Je kan ook select boxen aanmaken om bepaalde elementen weer te geven op grafieken of tabellen.

```
# display data
Value = { "carat": df['carat'],
        "price": df['price'],
        "depth": df['depth'],
        "table": df['table'] }

Selected2 = st.sidebar.selectbox("Kies Wat je wilt zien op de grafiek_1", list(Value.keys()),0)
#st.write(Value[Selected], use_column_width= True, caption= Value[Selected2])

st.area_chart(Value[Selected2])
```

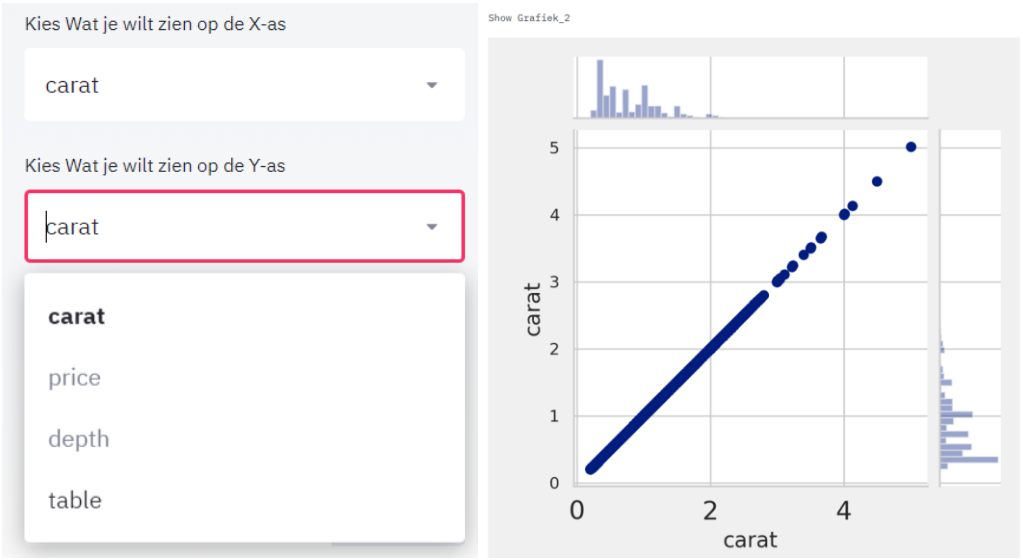
Output:



5) Met deze select boxen kan je ook de X- en Y-as naar keuze veranderen om verschillende grafieken te bekomen.

```
# display data
Value2 = { "carat": df['carat'],
        "price": df['price'],
        "depth": df['depth'],
        "table": df['table'] }

SelectedX = st.sidebar.selectbox("Kies Wat je wilt zien op de X-as", list(Value2.keys()),0)
#st.write(Value[Selected], use_column_width= True, caption= Value[Selected2])
SelectedY = st.sidebar.selectbox("Kies Wat je wilt zien op de Y-as", list(Value2.keys()),0)
#st.write(Value[Selected], use_column_width= True, caption= Value[Selected2])

st.text("Show Grafiek_2")
sns.jointplot(x= SelectedX , y= SelectedY , data=df , size=5)
st.pyplot()
```

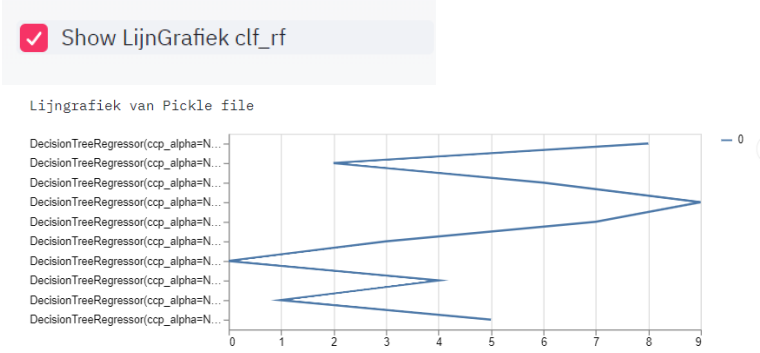
Output:



6) Als de applicatie te groot wordt kan je best gebruik maken van check boxen, dit betekend dat zolang de check box uit is deze niet getoond word in de applicatie

```
if st.sidebar.checkbox("Show LijnGrafiek clf_rf"):
    st.text("Lijngrafiek van Pickle file")
    st.line_chart(clf_rf)
```

Output:



7) Er is ook een mogelijkheid om afbeeldingen weer te geven in streamlit, alleen moet je er dan wel voor zorgen dat de afbellding online op een server staan want anders gaat hij ze niet kunnen weergeven. (we hebben dit getest met de lego blokken).

a) Een selectbox om een foto te kiezen.

```
# display picture
Fotos = { "02300BL": "http://11500991.pxl-ea-ict.be/SmartSys/02300%20BLUE/02300_BLUE_1.jpg",
        "3198LG": "http://11500991.pxl-ea-ict.be/SmartSys/3198%20LIGHTGREEN/3198_lightgreen_3.jpg",
        "3437BR": "http://11500991.pxl-ea-ict.be/SmartSys/3437%20BROWN/3437_brown_2.jpg",
        "3437GR": "http://11500991.pxl-ea-ict.be/SmartSys/3437%20GREEN/3437_green_1.jpg",
        "3437LB": "http://11500991.pxl-ea-ict.be/SmartSys/3437%20LIGHTBLUE/jason/3437_lightblue_5.jpg",
        "6437YE": "http://11500991.pxl-ea-ict.be/SmartSys/6474%20YELLOW/3437_yellow_3.jpg"}

Foto = st.sidebar.selectbox("Kies Foto", list(Fotos.keys()),0)
st.image(Fotos[Foto], use_column_width= True, caption= Fotos[Foto])
```

b) Dan kan je nog code schrijven als er een bepaalde foto geselecteerd is.

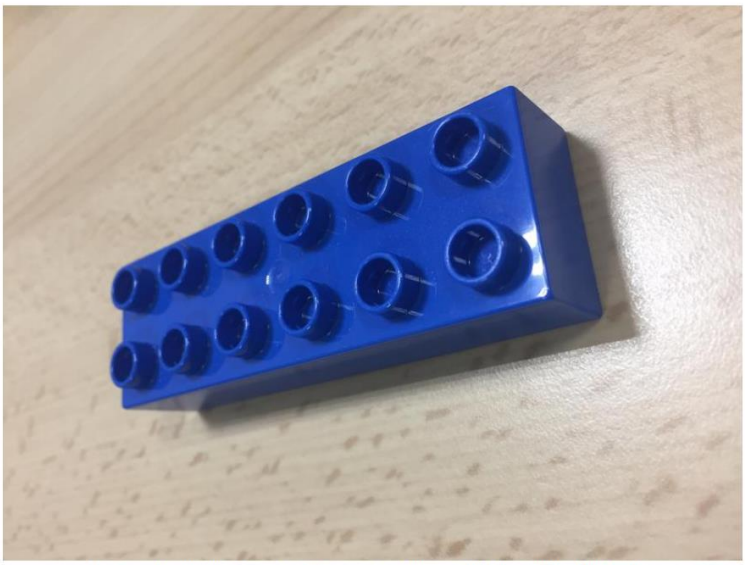
```
if Foto == "02300BL":
    st.write("Selected: 02300BL")
    st.write("Color: Blue")
if Foto == "3198LG":
    st.write("Selected: 3198LG")
    st.write("Color: Light Green")
if Foto == "3437BR":
    st.write("Selected: 3437BR")
    st.write("Color: Brown")
if Foto == "3437GR":
    st.write("Selected: 3437GR")
    st.write("Color: Green")
if Foto == "3437LB":
    st.write("Selected: 3437LB")
    st.write("Color: Light Blue")
if Foto == "6437YE":
    st.write("Selected: 6437YE")
    st.write("Color: Yellow")
```

Output:

a)

b)

Data Analyse van Lego met Streamlit



http://11500991.pxl-ea-ict.be/SmartSys/02300%20BLUE/02300_BLUE_1.jpg

Selected: 02300BL

Opmerkingen

1. Numbers als input kunnen voor errors zorgen