

# Medical Fraud Detection

Team Members:

Neel Shah  
Sagnik Ghosal  
Prerit Chaudhary  
Ishank Vasania



# Background and Use-case



Provider Fraud is one of the biggest problems facing Medicare. According to the government, the total Medicare spending increased exponentially due to frauds in Medicare claims. Healthcare fraud is an organized crime which involves peers of providers, physicians, beneficiaries acting together to make fraud claims.

We are trying to build an infrastructure for the Government to identify fraudulent providers and also get insights into frauds in specific facilities, by specific providers and in accordance with specific physicians. The Infrastructure will consists of 2 main parts:

1. Modelling - PyOD, sklearn, TensorFlow
2. Visualization - Dash, Bokeh

# Python Package Choices (Modelling)



- Python Library Name: **Scikit-learn**
- Author: **David Cournapeau**
- Release Date: **2007**
- Brief summary:
  - Open-source machine learning library with a diverse set of algorithms for classification, regression, clustering, and dimensionality reduction.
  - Provides data preprocessing, model selection, and evaluation tools.
  - Suitable for smaller datasets and models that do not require deep learning techniques.
  - Scikit-learn is simple to use and has a consistent API.

# Python Package Choices (Modelling)




- Python Library Name: **TensorFlow**
- TensorFlow was developed and is maintained by the **Google Brain team**.
- Release Date: **2017**
- Brief summary:
  - Provides a powerful and flexible platform for building and training various types of DL models
  - Has a rich set of low-level and high-level APIs allowing fine-grained control and ease of use.
  - Comprises a repository of pre-trained models that can be easily integrated into new projects.

# Python Package Choices (Modelling)



- Python Library Name: **PyOD (Python Outlier Detection)**
- Authors: **Yue Zhao, Zain Nasrullah, Zheng Li**
- Release Date: **2019**
- Brief summary:
  - Includes more than 40 anomaly detection algorithms, both supervised, and unsupervised
  - Highly flexible and customizable, with support for various data formats, feature types, and metrics.
  - Built on top of scikit-learn, and so it integrates seamlessly with scikit-learn's tools
  - Supports distributed computing, which allows users to scale to large and high-dimensional datasets
  - Is actively maintained and updated with new algorithms and features.

# Package Comparison (Modelling)



Feature/Functionality	scikit-learn	TensorFlow	PyOD
Type of Models	Traditional ML	Deep Learning	Traditional ML + SOTA DL
Data Preprocessing	Yes	Yes	Yes
Feature Selection	Yes	Yes	Yes
Supervised Learning	Yes	Yes	Yes
Unsupervised Learning	Yes	Yes	Yes
Semi-Supervised Learning	Yes	Yes	Yes
Time Series Analysis	Yes	Yes	Yes
Anomaly Detection	Yes (less extensive)	Yes (limited to only DL models)	Yes (extensive - both ML and DL)
Distributed Computing	No	Yes	Yes

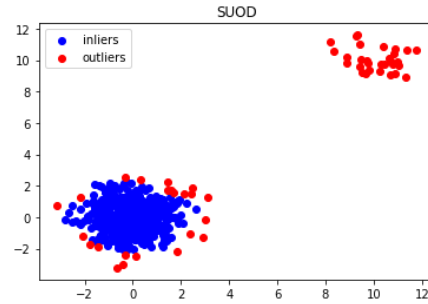
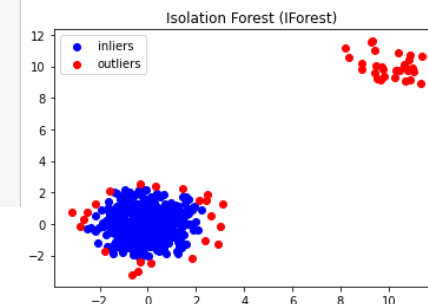
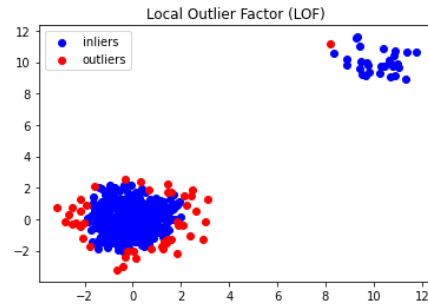
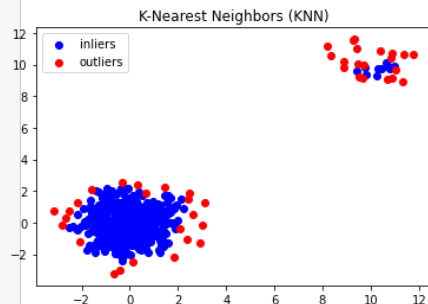
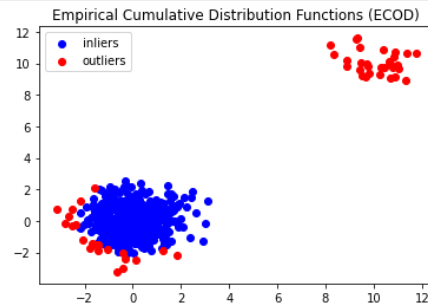
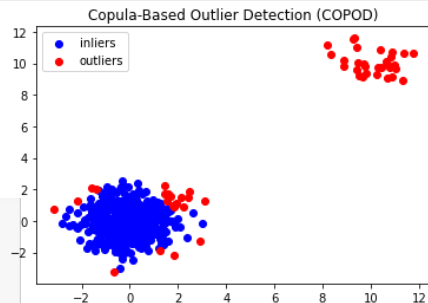
# Choice (Modelling)



- We have selected the PyOD package for the modelling aspect in our project.
- Why:
  - PyOD provides a more extensive collection of anomaly detection models, including supervised, unsupervised, ML-based and DL-based.
  - The implementation is very easy (2-3 lines of code)

# PyOD Demo

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from pyod.models.ecod import ECOD
4 from pyod.models.copod import COPOD
5 from pyod.models.knn import KNN
6 from pyod.models.lof import LOF
7 from pyod.models.iforest import IForest
8 from pyod.models.suod import SUOD
9
10 inliers = np.random.multivariate_normal(mean=[0, 0], cov=[[1, 0], [0, 1]], size=500)
11 outliers = np.random.multivariate_normal(mean=[10, 10], cov=[[1, 0], [0, 1]], size=30)
12 X = np.vstack([inliers, outliers])
13 y = np.concatenate([np.zeros(len(inliers)), np.ones(len(outliers))])
14
15
16 models = {"Copula-Based Outlier Detection (COPOD)": COPOD(contamination=0.1),
17           "Empirical Cumulative Distribution Functions (ECOD)": ECOD(contamination=0.1),
18           "K-Nearest Neighbors (KNN)": KNN(contamination=0.1),
19           "Local Outlier Factor (LOF)": LOF(contamination=0.1),
20           "Isolation Forest (IForest)": IForest(n_estimators=100),
21           "SUOD": SUOD(contamination=0.1)}
22
23
24 for model_name, model in models.items():
25     model.fit(X)
26     y_pred = model.predict(X)
27
28     # Separate the inliers and outliers
29     inlier_indices = np.where(y_pred == 0)[0]
30     outlier_indices = np.where(y_pred == 1)[0]
31
32     # Visualize the results
33     plt.scatter(X[inlier_indices, 0], X[inlier_indices, 1], c='b', marker='o', label='inliers')
34     plt.scatter(X[outlier_indices, 0], X[outlier_indices, 1], c='r', marker='o', label='outliers')
35     plt.legend()
36     plt.title(model_name)
37     plt.show()
```





# Drawbacks/Remaining Concerns (Modelling)



- PyOD:
  - PyOD mainly relies on traditional ML techniques for anomaly detection. While it has a few neural network-based models, support for DL models is currently limited.
  - Some of the models in PyOD can be computationally expensive to train. This can be a disadvantage when dealing with large datasets.
  - PyOD is a relatively new library compared to more established machine learning libraries like scikit-learn. It may not have as much documentation or community support available, which can make it more challenging to use for beginners.

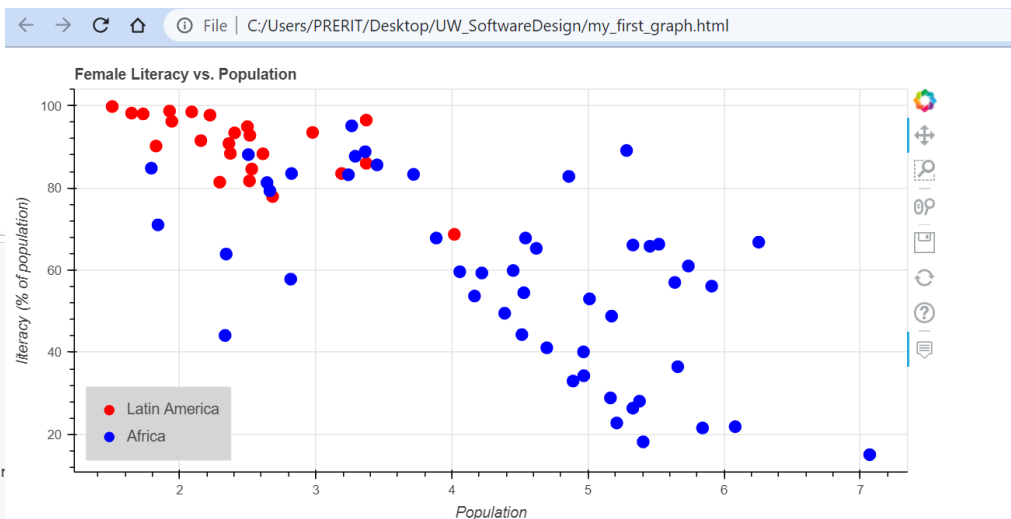
# Python Package Choices (Visualization)



- Python Library Name: **Bokeh**
- Author and Release Date : **Joseph Cottam and team (2014)**
- **Brief summary:**
  - Bokeh is a Python interactive visualization library that targets modern web browsers for presentation providing elegant, concise construction of novel graphics with high-performance interactivity over very large or streaming datasets in a quick and easy way. Bokeh exposes different interface levels to the users:
    - Allows plotting centered around composing visual glyphs that provides the most flexibility to application developers.
    - Allows interface that can be used to build complex statistical plots as quickly and as simply as possible.

# Bokeh Demo

```
In [22]: import pandas as pd
from pprint import pprint as pp
from itertools import combinations
from pathlib import Path
import requests
import numpy as np
from bokeh.io import output_notebook, curdoc
from bokeh.plotting import figure, show
from bokeh.sampledata.iris import flowers
from bokeh.sampledata.iris import flowers as iris_df
from bokeh.models import ColumnDataSource, HoverTool, CategoricalColorMapper, Slider
from bokeh.models import CheckboxGroup, RadioGroup, Toggle, Button
from bokeh.models.widgets import Tabs, Panel
from bokeh.layouts import row, column, gridplot, widgetbox
from bokeh.palettes import Spectral6
from bokeh.themes import Theme
import yaml
from bokeh.plotting import figure, output_file
latin_america = ColumnDataSource(lit[lit.Continent == 'LAT'])
africa = ColumnDataSource(lit[lit.Continent == 'AF'])
p = figure(plot_height=400, plot_width=800, title='Female Literacy vs. Population', x_axis_label='Population', y_axis_label='Literacy (% of population)')
p.circle('fertility', 'female literacy', source=latin_america, size=10, color='red', legend_label='Latin America')
p.circle('fertility', 'female literacy', source=africa, size=10, color='blue', legend_label='Africa')
p.legend.location = 'bottom_left'
p.legend.background_fill_color = 'lightgray'
hover = HoverTool(tooltips=[('Country', '@Country')])
p.add_tools(hover)
output_file('my_first_graph.html')
show(p)
```



# Python Package Choices (Visualization)



- Python Library Name: **Dash**
- Author and Release Date: **Chris Parmer and the team at Plotly (2017)**
- Brief summary:
  - Dash is an open-source framework for building web applications with Python. It allows developers to create interactive, data-driven dashboards and web applications with minimal coding.
  - Dash is built on top of Flask, Plotly.js, and React.js, and provides a simple and flexible way to create and customize web applications. With Dash, developers can build dynamic, responsive, and real-time web applications that can display live data, interactive visualizations, and complex data workflows.

# Dash Demo

```
In [*]: from dash import Dash, html, dcc
import plotly.express as px
import pandas as pd

app = Dash(__name__)

# assume you have a "Long-form" data frame
# see https://plotly.com/python/px-arguments/ for more options
df = pd.DataFrame({
    "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
    "Amount": [4, 1, 2, 2, 4, 5],
    "City": ["SF", "SF", "SF", "Montreal", "Montreal", "Montreal"]
})

fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

app.layout = html.Div(children=[
    html.H1(children="Hello Dash"),

    html.Div(children='''
        Dash: A web application framework for your data.
    '''),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
])

if __name__ == '__main__':
    app.run_server(debug=True, use_reloader=False)
```

Dash is running on <http://127.0.0.1:8050/>

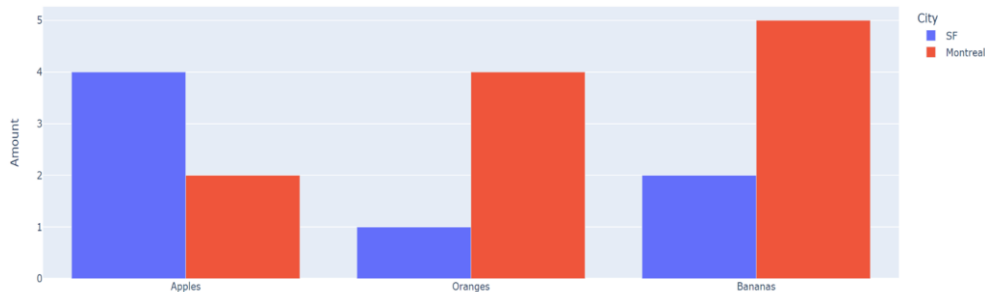
Dash is running on <http://127.0.0.1:8050/>

```
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
```

127.0.0.1:8050

## Hello Dash

Dash: A web application framework for your data.



# Package Comparison (Visualization)



Feature	Bokeh	Dash
Main focus	Visualization	Web applications
Approach	Declarative	Imperative
Language	Python	Python
Interactive plotting	Yes	Yes
Dashboards	Yes	Yes
Tabular data support	Yes, with the DataTable widget	Yes, with the DataTable component
Integration with other libraries	Yes, can be used with Pandas and NumPy	Yes, can be used with Pandas and NumPy
Server requirements	Bokeh server required for some features	Dash can be used with any web server
Customization	Extensible with custom models	Customizable with Dash components and CSS
Learning curve	Moderate	Moderate

# Choice (Visualization)



- We are still exploring Bokeh and Dash for the visualization aspect in our project as both have very similar functionalities.
- Why:
  - Bokeh handles large datasets well as it has a custom storage class called ColumnDataSource which imitates the functionality between a pandas DataFrame and a dict. It allows us to implement interactive visualizations and tools such as buttons, sliders, radio buttons, dropdowns, text input, tables.
  - Dash is also web based visualization tool and can be easily deployed with Python. It doesn't need any web server and also provides interactive functionalities

# Drawbacks/Remaining Concerns (Visualization)



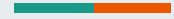
- **Bokeh:**
  - Bokeh is a library that somewhat has a middle-level interface, it often takes less code than Matplotlib but takes more code to produce the same plot as Seaborn, Altair, or Plotly.



# References



- PyOD - <https://pyod.readthedocs.io/en/latest/>
- Sklearn - [https://scikit-learn.org/stable/modules/outlier\\_detection.html](https://scikit-learn.org/stable/modules/outlier_detection.html)
- TensorFlow - <https://www.tensorflow.org/>
- Bokeh - <https://bokeh.org/>
- Dash - <https://plotly.com/dash/>



# Questions?