Using Survey Data as a Predictor of Pandemic Vaccination

Jupyter Dash Results Dashboard

Mark Patterson, March 2021

Displaying Classificaion Modeling Results

This interactive dashboard allows a user to select between 3 different data preperations, and see the associated model results for 6 models. This is a work in progress and I hope to add on additional elements to display, for e other parts of the project and an EDA chart explorer.

Import the relevant libraries and data

In [1]:
```
from jupyter_dash import JupyterDash
```

In [2]:
```
# Import the libraries
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.express as px
from jupyter_dash import JupyterDash
from dash.dependencies import Input, Output
```

In [4]:
```
# When running in JupyterHub or Binder, call the infer_jupyter_config function to detect the proxy configuration.
JupyterDash.infer_jupyter_proxy_config()
```

In [ ]:
```
```

Creating the tables from classification modeling results

In [3]:
```
import dash_table
```

In [4]:
```
# Base modeling
data1 = [['XGBoost', 0.85, 0.68], ['Random Forest', 0.84, 0.68], ['SVC', 0.84, 0.67], ['Logistic Regression', 0.84, 0.66], ['KNN', 0.81, 0.54], ['Decision Trees', 0.7
dfA_results = pd.DataFrame(data1, columns = ['Classification_model', 'Accuracy', 'Precision1'])
dfA_results
```

Out[4]:

	Classification_model	Accuracy	Precision1
0	XGBoost	0.85	0.68
1	Random Forest	0.84	0.68
2	SVC	0.84	0.67
3	Logistic Regression	0.84	0.66
4	KNN	0.81	0.54
5	Decision Trees	0.75	0.40

In [5]:
```
# Base modeling with SMOTE
data2 = [['XGBoost', 0.83, 0.61], ['Random Forest', 0.83, 0.64], ['SVC', 0.80, 0.52], ['Logistic Regression', 0.77, 0.48], ['KNN', 0.67, 0.35], ['Decision Trees', 0.7
dfB_results = pd.DataFrame(data2, columns = ['Classification_model', 'Accuracy', 'Precision1'])
dfB_results
```

Out[5]:

	Classification_model	Accuracy	Precision1
0	XGBoost	0.83	0.61
1	Random Forest	0.83	0.64
2	SVC	0.80	0.52
3	Logistic Regression	0.77	0.48
4	KNN	0.67	0.35
5	Decision Trees	0.74	0.41

In [6]:
```
# Approach B - Base modeling
data3 = [['XGBoost', 0.84, 0.71], ['Random Forest', 0.84, 0.72], ['SVC', 0.83, 0.70], ['Logistic Regression', 0.83, 0.69], ['KNN', 0.80, 0.59], ['Decision Trees', 0.7
dfC_results = pd.DataFrame(data3, columns = ['Classification_model', 'Accuracy', 'Precision1'])
dfC_results
```

Out[6]:

	Classification_model	Accuracy	Precision1
0	XGBoost	0.84	0.71
1	Random Forest	0.84	0.72
2	SVC	0.83	0.70
3	Logistic Regression	0.83	0.69
4	KNN	0.80	0.59
5	Decision Trees	0.76	0.45

Creating the code to display the dashboard in this notebook

In [8]:
```
# One basic table - for modeling results
# Build App
external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']
app = JupyterDash(__name__, external_stylesheets=external_stylesheets)

Table1 = dash_table.DataTable(
    id='tableA',
    columns=[{"name": i, "id": i} for i in dfA_results.columns],
    data=dfA_results.to_dict('records'),
    style_cell=dict(textAlign='left'),
        style_header=dict(backgroundColor="paleturquoise"),
        style_data=dict(backgroundColor="lavender")
)
Table2 = dash_table.DataTable(
    id='tableB',
    columns=[{"name": i, "id": i} for i in dfB_results.columns],
    data=dfB_results.to_dict('records'),
    style_cell=dict(textAlign='left'),
        style_header=dict(backgroundColor="paleturquoise"),
        style_data=dict(backgroundColor="lavender")
)

Table3 = dash_table.DataTable(
    id='tableC',
    columns=[{"name": i, "id": i} for i in dfC_results.columns],
    data=dfC_results.to_dict('records'),
    style_cell=dict(textAlign='left'),
        style_header=dict(backgroundColor="paleturquoise"),
        style_data=dict(backgroundColor="lavender")
)

# This is the layout of the dashboard
app.layout = html.Div([
    html.H2("Classification Model Results"),
    html.Label([
        "Select data preperation to see results of 6 models",
        dcc.Dropdown(
            id='column-dropdown', clearable=False,
            value = 'A-Basic-preprocessing', options=[
                {'label': c, 'value': c}
                for c in ['A-Basic-preprocessing', 'B-Basic-with-SMOTE', 'C-One-Hot-Encoded']
            ])

    ]),
    html.Div(id='results_table', style={'padding': 20})
])


# Define callback to update graph
@app.callback(
    Output('results_table', 'children'),
    [Input("column-dropdown", "value")]
)
def update_figure(model):
    if model == 'A-Basic-preprocessing':
        return Table1
    if model == 'B-Basic-with-SMOTE':
        return Table2
    if model == 'C-One-Hot-Encoded':
        return Table3

# Run app and display result inline in the notebook
app.run_server(mode='inline')
```

Classification Model Results

Select data preperation to see results of 6 models

C-One-Hot-Encoded

Classification_model	Accuracy	Precision1
XGBoost	0.84	0.71
Random Forest	0.84	0.72
SVC	0.83	0.7
Logistic Regression	0.83	0.69
KNN	0.8	0.59
Decision Trees	0.76	0.45

Observations on the dashboard

This is really just a proof of concept at this point. There are some changes and enhancements I would like to make, including:

- figure out why excess whitespace is included below the table
- add a title and description above the table - this would need to change with each table
- better control the width of the table columns - I know how to change the proportion, but also need to change the overall size of the table
- improve the size and font display
- Change the color fill within the table