In [1]:

```
import dash
from dash import html, dcc
from dash.dependencies import Input, Output, State
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

data = pd.read_csv("C:\\Users\\lexiw\\OneDrive\\Desktop\\DSC 410\\churn-bigml-80.csv")

# Needed to one hot encode the states for the model to work
data = pd.get_dummies(data, columns=['State', 'International plan', 'Voice mail plan'], drop_first=False)

# Split for target and features; and training /testing set
X = data.drop(columns=['Churn'])
y = data['Churn'].astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model training
model = RandomForestClassifier(max_depth=20, n_estimators=200, random_state=42)
model.fit(X_train, y_train)

# Dash start/ layout/ dropdown better than text typing, text typing was giving formatting errors

app = dash.Dash(__name__)
app.layout = html.Div([
    html.H1("Customer Churn Prediction"),
    dcc.Dropdown(
        id='input-state',
        options=[{'label': i.split('_')[-1], 'value': i} for i in X.columns if i.startswith('State_')],
        placeholder='Select a state',
    ),
    dcc.Dropdown(
        id='input-international-plan',
        options=[
            {'label': 'Yes', 'value': 'International plan_Yes'},
            {'label': 'No', 'value': 'International plan_No'}
        ],
        placeholder='International Plan'
    ),
    dcc.Dropdown(
        id='input-voice-mail-plan',
        options=[
            {'label': 'Yes', 'value': 'Voice mail plan_Yes'},
```

```
            {'label': 'No', 'value': 'Voice mail plan_No'}
        ],
        placeholder='Voice Mail Plan'
    ),
    html.Button('Predict Churn', id='predict-button', n_clicks=0),
    html.Div(id='prediction-output')
])

@app.callback(
    Output('prediction-output', 'children'),
    Input('predict-button', 'n_clicks'),
    [State('input-state', 'value'),
     State('input-international-plan', 'value'),
     State('input-voice-mail-plan', 'value')]
)
def update_output(n_clicks, state, international_plan, voice_mail_plan):
    if n_clicks > 0:
        try:
            # Preparing for input for predictions, and making predictions
            input_data = {col: [0] for col in X_train.columns}
            if state:
                input_data[state] = [1]
            if international_plan:
                input_data[international_plan] = [1]
            if voice_mail_plan:
                input_data[voice_mail_plan] = [1]

            input_df = pd.DataFrame(input_data)

            prediction = model.predict(input_df)
            return f'Prediction: {"Churn Possibility High" if prediction[0] == 1 else "Churn Possibility Low"}
        except Exception as e:
            return f'Error processing input: {str(e)}'
    return 'Enter values and click predict.'

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

Customer Churn Prediction

Select a state

International Plan

Voice Mail Plan

Predict Churn

Enter values and click predict.

In [ ]: