Example of Conversation with GPT Making A Graph Showing the Best Possible Combinations of False Negatives and False Positives, Varying Model Parameters

Me

I'm trying to show trade offs between false negatives and false positives. I have three variables I vary in my model: small vs. large, count (1, 100, or 1000) and tolerance (.4, .5, .6, .7, .8, .9). I have two tables: one with rate of misidentification among true positives, one with rate of misidentification among true negatives, by my varying variables. I also have line charts. How can I show in a chart this trade off between false positives and negatives?

ChatGPT

One common way to visualize the trade-off between false positives and false negatives is through a Receiver Operating Characteristic (ROC) curve. However, ROC curves typically show the trade-off between true positive rate (sensitivity) and false positive rate (1-specificity) for different threshold values. If you have different models (or different configurations of a model) rather than different thresholds, you might need a different approach.

Given your description, it sounds like you have a multi-dimensional problem. You have three variables you're varying (size, count, and tolerance), and two outcomes you're interested in (misidentification among true positives, and misidentification among true negatives). This is a complex space to visualize!

One approach could be to create a series of scatter plots, where each plot represents one combination of size and count. On each plot, the x-axis could represent tolerance, and the y-axis could represent the rate of misidentification. You could then plot two series on each plot: one for the rate of misidentification among true positives (false negatives), and one for the rate of misidentification among true negatives (false positives). This would allow you to see how the trade-off between these two rates changes as you vary the tolerance, for each combination of size and count.

Another approach could be to use a 3D scatter plot or surface plot, where the x, y, and z axes represent tolerance, misidentification among true positives, and misidentification among true negatives, respectively. Each point on the plot would then represent one combination of size, count, and tolerance. However, this can be harder to interpret, especially if you're trying to include all combinations of size and count on the same plot.

Finally, you could consider using a heatmap or contour plot to represent this data. In this case, you could create a grid where the x and y axes represent two of your variables (e.g., size and count), and the color of each cell represents the rate of misidentification (either true positives or true negatives). You could then create two such plots, one for each type of misidentification, and compare them side by side.

Remember, the best way to visualize your data will depend on the specific nature of your data and the message you're trying to convey. It may be helpful to try out a few different approaches and see which one works best for your needs.

Me

I'm interested in something that looks like a ppf or utility curve? I think we could do that for each of the lines that represents a permutation of the three variables

ChatGPT

A Production Possibility Frontier (PPF) or utility curve could indeed be a useful way to visualize the trade-off between false positives and false negatives. In this case, one axis would represent the rate of false positives and the other axis would represent the rate of false negatives. Each point on the curve would represent a different configuration of your model, with the understanding that moving along the curve involves a trade-off between false positives and false negatives.

To create such a plot, you would first need to calculate the rate of false positives and false negatives for each configuration of your model. Then, for each configuration, you would plot a point where the x-coordinate is the rate of false positives and the y-coordinate is the rate of false negatives. You could use different colors or symbols to represent different values of your three variables (size, count, and tolerance).

However, keep in mind that a PPF or utility curve typically assumes that all else is equal, and that you can freely trade off between the two quantities being plotted. In your case, this would mean assuming that you can freely adjust the rate of false positives and false negatives by changing the configuration of your model. This may or may not be a valid assumption, depending on the specifics of your model and data.

Also, note that a PPF or utility curve typically represents the "best possible" trade-offs. If your model configurations do not all lie on a single curve, or if there are some configurations that are clearly worse than others (i.e., they have a higher rate of both false positives and false negatives), then this type of plot may not be the most appropriate. In such cases, a scatter plot or other type of plot might be more informative.

Me

I think I want to try this. Can you write me Python code? I have two tables as inputs, df_pos and df_neg. Columns for both are small_1, small_10, small_100, large_1, large_10, large_100. Index is .4, .5, .6, .7,.8,.9. I like plotly

ChatGPT

Sure, I can provide you with a basic example of how you might create a scatter plot using Plotly in Python. This example assumes that `df_pos` and `df_neg` are pandas DataFrames and that the

columns represent different configurations of your model, while the index represents the tolerance.

Here's a basic example:

python Copy code

```python
import plotly.graph_objects as go

# Create a new figure
fig = go.Figure()

# Iterate over the columns in df_pos and df_neg
for col in df_pos.columns:
    # Add a scatter plot to the figure for each column
    fig.add_trace(go.Scatter(
        x=df_neg[col],  # false positive rate
        y=df_pos[col],  # false negative rate
        mode='lines+markers',
        name=col
    ))

# Set the title and labels
fig.update_layout(
    title='Trade-off between False Positives and False Negatives',
    xaxis_title='False Positive Rate',
    yaxis_title='False Negative Rate',
)

# Show the figure
fig.show()
```

This will create a scatter plot where each point represents a different tolerance value, and each series (line) represents a different configuration of your model. The x-coordinate of each point is the false positive rate, and the y-coordinate is the false negative rate.

Please note that this is a basic example and you might need to adjust it to fit your specific needs. For example, you might want to add more information to the plot, adjust the colors or symbols used for each series, or change the scale of the axes. You can find more information on how to customize your plot in the Plotly documentation.

Me

Can you show me how to add the index info from the original dfs to the hovertext?

ChatGPT

Sure, you can add hover text to each data point in a Plotly scatter plot by using the `hovertext` parameter. Here's how you can modify the previous code to include the index values from your original DataFrames as hover text:

In this code, `hover_text` is a list of strings that is created by iterating over the index of `df_pos` (which should be the same as the index of `df_neg` if your DataFrames are structured as you described). Each string in `hover_text` is formatted to include the corresponding tolerance value. This list is then passed to the `hovertext` parameter of `go.Scatter` to set the hover text for each data point.

python Copy code

```python
import plotly.graph_objects as go

# Create a new figure
fig = go.Figure()

# Iterate over the columns in df_pos and df_neg
for col in df_pos.columns:
    # Create hover text for each data point
    hover_text = ['tolerance: {}'.format(i) for i in df_pos.index]

    # Add a scatter plot to the figure for each column
    fig.add_trace(go.Scatter(
        x=df_neg[col],  # false positive rate
        y=df_pos[col],  # false negative rate
        mode='lines+markers',
        name=col,
        hovertext=hover_text,
    ))

# Set the title and labels
fig.update_layout(
    title='Trade-off between False Positives and False Negatives',
    xaxis_title='False Positive Rate',
    yaxis_title='False Negative Rate',
)

# Show the figure
fig.show()
```

Regenerate response

Me

Can I just show the best options, and use the hovertext to show what the model is? It should just be one curve.

ChatGPT

If you want to show only the best options and create a single curve, you'll need to define what "best" means in your context. Typically, this might mean minimizing both false positives and false negatives, but the specific trade-off between these two will depend on your specific application.

One common approach is to choose the model configuration that minimizes the total cost, where the cost is a weighted sum of the false positive rate and the false negative rate. The weights in this sum would reflect the relative importance of minimizing false positives versus false negatives.

Once you've defined your cost function, you can calculate the cost for each model configuration, choose the configuration with the lowest cost for each tolerance level, and plot these on your graph.

Here's an example of how you might do this:

python Copy code

```python
import plotly.graph_objects as go
import pandas as pd

# Define the weights for the cost function
weight_fp = 1  # weight for false positives
weight_fn = 1  # weight for false negatives

# Calculate the cost for each model configuration
cost = weight_fp * df_neg + weight_fn * df_pos

# Find the configuration with the lowest cost for each tolerance level
best_config = cost.idxmin(axis=1)

# Create DataFrames for the best false positive and false negative rates
best_fp = pd.DataFrame(df_neg.lookup(df_neg.index, best_config), index=df_ne
best_fn = pd.DataFrame(df_pos.lookup(df_pos.index, best_config), index=df_po

# Create a new figure
fig = go.Figure()

# Add a scatter plot to the figure for the best configuration
fig.add_trace(go.Scatter(
    x=best_fp['Best'],  # best false positive rate
    y=best_fn['Best'],  # best false negative rate
    mode='lines+markers',
    name='Best Configuration',
    hovertext=best_config,
))

# Set the title and labels
fig.update_layout(
    title='Trade-off between False Positives and False Negatives',
    xaxis_title='False Positive Rate',
    yaxis_title='False Negative Rate',
)

# Show the figure
fig.show()
```

In this code, `weight_fp` and `weight_fn` are the weights for the false positive rate and false negative rate, respectively, in the cost function. You should adjust these weights based on the relative importance of minimizing false positives versus false negatives in your application.

The `cost` DataFrame is calculated by multiplying the false positive and false negative rates by their respective weights and adding the results. The `best_config` Series is then created by finding the column name (i.e., the model configuration) with the lowest cost for each tolerance level.

The `best_fp` and `best_fn` DataFrames are created by looking up the best false positive and false negative rates for each tolerance level, respectively. These are then plotted on the graph, with the best configuration as hover text.

Me

No, sorry, I want to show the curve that's the lowest point ? Like for each rate of false positive, the point that's the lowest rate of false negatives

ChatGPT

Ah, I see. If you want to plot the curve that represents the lowest false negative rate for each false positive rate, you can do so by finding the minimum false negative rate for each unique false positive rate. Here's how you can do it:

```python
import plotly.graph_objects as go
import pandas as pd

# Reshape the dataframes to long format
df_pos_long = df_pos.melt(var_name='model', value_name='fn_rate')
df_neg_long = df_neg.melt(var_name='model', value_name='fp_rate')

# Merge the dataframes
df = pd.concat([df_pos_long, df_neg_long['fp_rate']], axis=1)

# Group by false positive rate and find the minimum false negative rate
df_min = df.groupby('fp_rate')['fn_rate'].min().reset_index()

# Create a new figure
fig = go.Figure()

# Add a scatter plot to the figure for the best configuration
fig.add_trace(go.Scatter(
    x=df_min['fp_rate'],  # best false positive rate
    y=df_min['fn_rate'],  # best false negative rate
    mode='lines+markers',
    name='Best Configuration',
))

# Set the title and labels
fig.update_layout(
    title='Trade-off between False Positives and False Negatives',
    xaxis_title='False Positive Rate',
    yaxis_title='False Negative Rate',
)

# Show the figure
fig.show()
```

In this code, the `melt` function is used to reshape the dataframes from wide format to long format. Then, the dataframes are merged into a single dataframe. The `groupby` function is used to group the data by false positive rate, and the `min` function is used to find the minimum false negative rate for each group. The resulting dataframe, `df_min`, contains the minimum false negative rate for each unique false positive rate, which is then plotted on the graph.

Me

Thanks. Can you make the hovertext show the original info about the model - small vs large, count, tolerance?

ChatGPT

Sure, you can add the original model information to the hover text. However, since you're taking the minimum false negative rate for each unique false positive rate, there might be multiple model configurations that achieve this minimum rate. In this case, you'll need to decide how to handle these multiple configurations. One option is to list all configurations that achieve the minimum rate in the hover text.

Here's how you can modify the previous code to include the model information in the hover text:

python Copy code

```python
import plotly.graph_objects as go
import pandas as pd

# Reshape the dataframes to long format
df_pos_long = df_pos.reset_index().melt(id_vars='index', var_name='model', v
df_neg_long = df_neg.reset_index().melt(id_vars='index', var_name='model', v

# Merge the dataframes
df = pd.concat([df_pos_long, df_neg_long['fp_rate']], axis=1)

# Group by false positive rate and find the minimum false negative rate
df_min = df.groupby('fp_rate').apply(lambda x: x.loc[x['fn_rate'].idxmin()])

# Create a new figure
fig = go.Figure()

# Add a scatter plot to the figure for the best configuration
fig.add_trace(go.Scatter(
    x=df_min['fp_rate'],  # best false positive rate
    y=df_min['fn_rate'],  # best false negative rate
    mode='lines+markers',
    name='Best Configuration',
    hovertext=df_min['model'] + ', tolerance: ' + df_min['index'].astype(str
))

# Set the title and labels
fig.update_layout(
    title='Trade-off between False Positives and False Negatives',
    xaxis_title='False Positive Rate',
    yaxis_title='False Negative Rate',
)

# Show the figure
fig.show()
```

In this code, the `reset_index` function is used to include the tolerance level (which was previously the index) as a column in the dataframe. Then, in the `groupby` function, a lambda function is used to return the entire row (including the model and tolerance information) that has the minimum false negative rate for each group. The `hovertext` parameter of `go.Scatter` is then set to a string that includes the model and tolerance information.