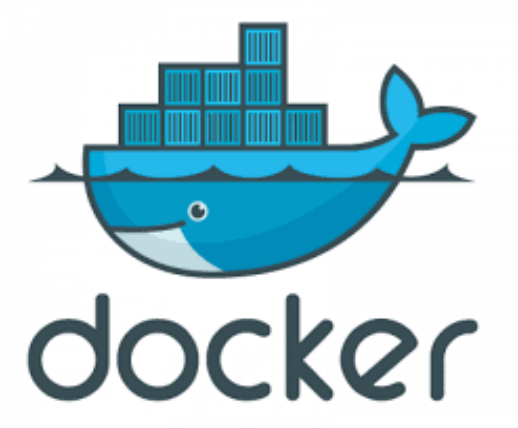
Building Web Apps with Python

Why Build a Web App?

- To change the *World!*
- To better understand how the model works.
- To generate more insights.
- To attract more audience and make a more compelling case.
- To share your model with others.

Available Tools

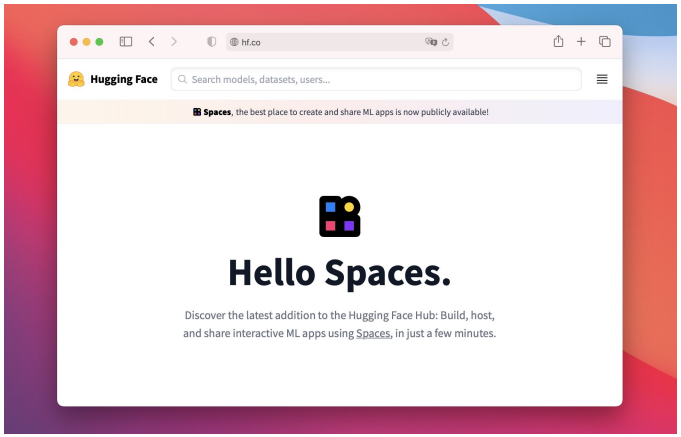


Streamlit

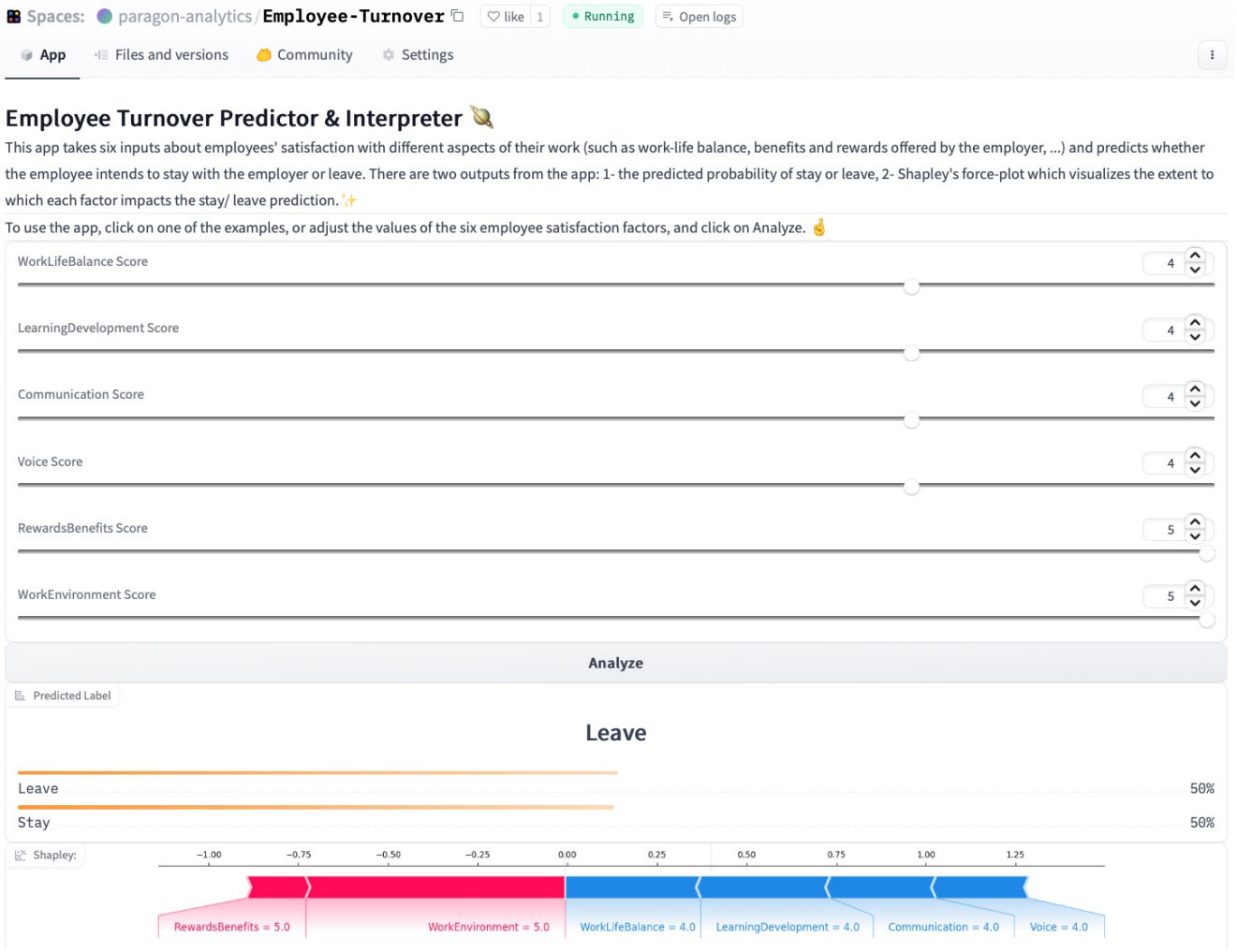
Where to Host the App?

- Host the app on a local machine
- Host the app on cloud
  - AWS
  - GCP
  - HuggingFace
  - Other

aws marketplace



gradio


Spaces: paragon-analytics / Employee-Turnover
like 1
Running
Open logs
App
Files and versions
Community
Settings
Employee Turnover Predictor & Interpreter
This app takes six inputs about employees' satisfaction with different aspects of their work (such as work-life balance, benefits and rewards offered by the employer, ...) and predicts whether the employee intends to stay with the employer or leave. There are two outputs from the app: 1- the predicted probability of stay or leave, 2- Shapley's force-plot which visualizes the extent to which each factor impacts the stay/ leave prediction.
To use the app, click on one of the examples, or adjust the values of the six employee satisfaction factors, and click on Analyze.
WorkLifeBalance Score
4
LearningDevelopment Score
4
Communication Score
4
Voice Score
4
RewardsBenefits Score
5
WorkEnvironment Score
5
Analyze
Predicted Label
Leave
Leave
50%
Stay
50%
Shapley:
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
RewardsBenefits = 5.0
WorkEnvironment = 5.0
WorkLifeBalance = 4.0
LearningDevelopment = 4.0
Communication = 4.0
Voice = 4.0

Gradio- Hello World!

Import Gradio

```
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

demo = gr.Interface(fn=greet, inputs="text", outputs="text")

demo.launch()
```

Create a function that takes an input and returns an output

Create the user interface

Launch the app

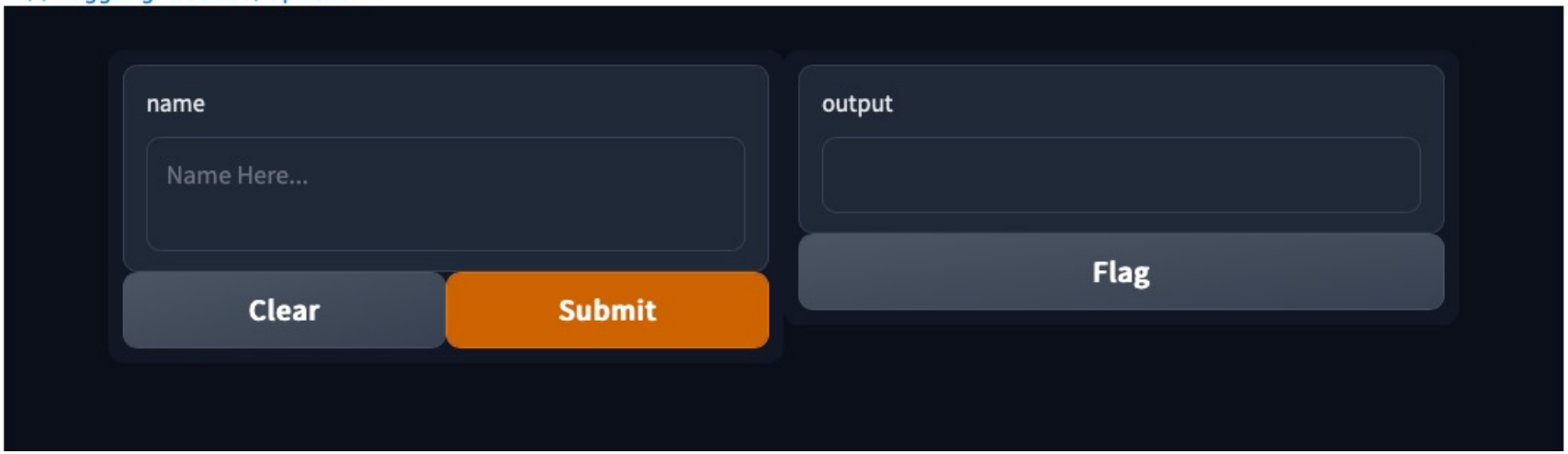
The Interface Class

The core **`Interface`** class is initialized with three required parameters:

- **`fn`**: the function to wrap a UI around
- **`inputs`**: which component(s) to use for the input (e.g. **`"text"`**, **`"image"`** or **`"audio"`**)
- **`outputs`**: which component(s) to use for the output (e.g. **`"text"`**, **`"image"`** or **`"label"`**)

```python
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

demo = gr.Interface(
    fn=greet,
    inputs=gr.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text",
)
demo.launch(share = True)
```

```
Running on local URL:  http://127.0.0.1:7861
Running on public URL: https://4b7c915e1f42666c.gradio.app

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NEW!), check out Spaces: http
://huggingface.co/spaces
```



Interface- Inputs and Outputs

```
demo = gr.Interface(
    fn=greet,
    inputs=["text", "checkbox", gr.Slider(0, 100)],
    outputs=["text", "number"],
)
demo.launch()
```

Gradio Blocks

Gradio offers two classes to build apps:

1. **Interface**, that provides a high-level abstraction for creating demos that we've been discussing so far.

2. **Blocks**, a low-level API for designing web apps with more flexible layouts and data flows. Blocks allows you to do things like feature multiple data flows and demos, control where components appear on the page, handle complex data flows (e.g. outputs can serve as inputs to other functions), and update properties/visibility of components based on user interaction — still all in Python. If this customizability is what you need, try Blocks instead!

Hello, Blocks

```python
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output)

demo.launch()
```

Input

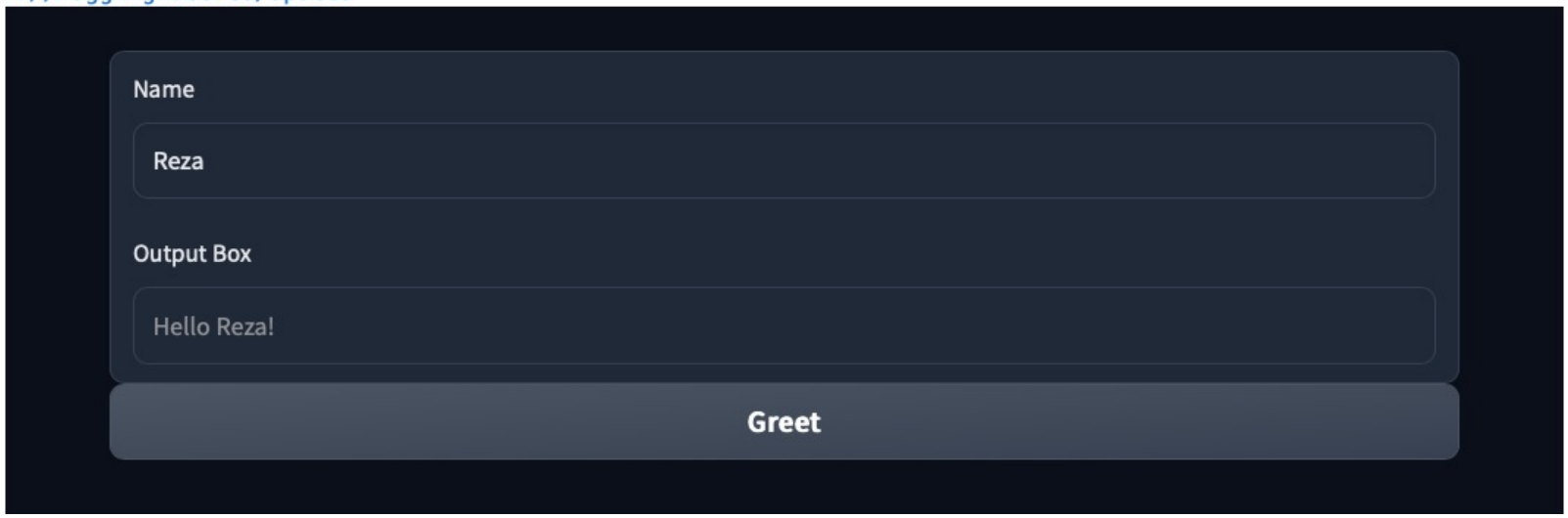
Output

Button

What we had in the interface

```python
import gradio as gr

def greet(name):
    return "Hello " + name + "!"

with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output)

demo.launch(share = True)
```

```
Running on local URL:  http://127.0.0.1:7862
Running on public URL: https://de2a1a408e5b37f5.gradio.app

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NEW!), check out Spaces: https://huggingface.co/spaces
```



Types of Inputs/ Outputs

gradio.Textbox(…)

gradio.Number(…)

gradio.Slider(…)

gradio.Checkbox(…)

gradio.Image(…)

gradio.Video(…)

gradio.File(…)

gradio.JSON(…)

gradio.HTML(…)

Types of Inputs/ Outputs

gradio.Textbox(…)

gradio.Number(…)

gradio.Slider(…)

gradio.Checkbox(…)

gradio.Image(…)

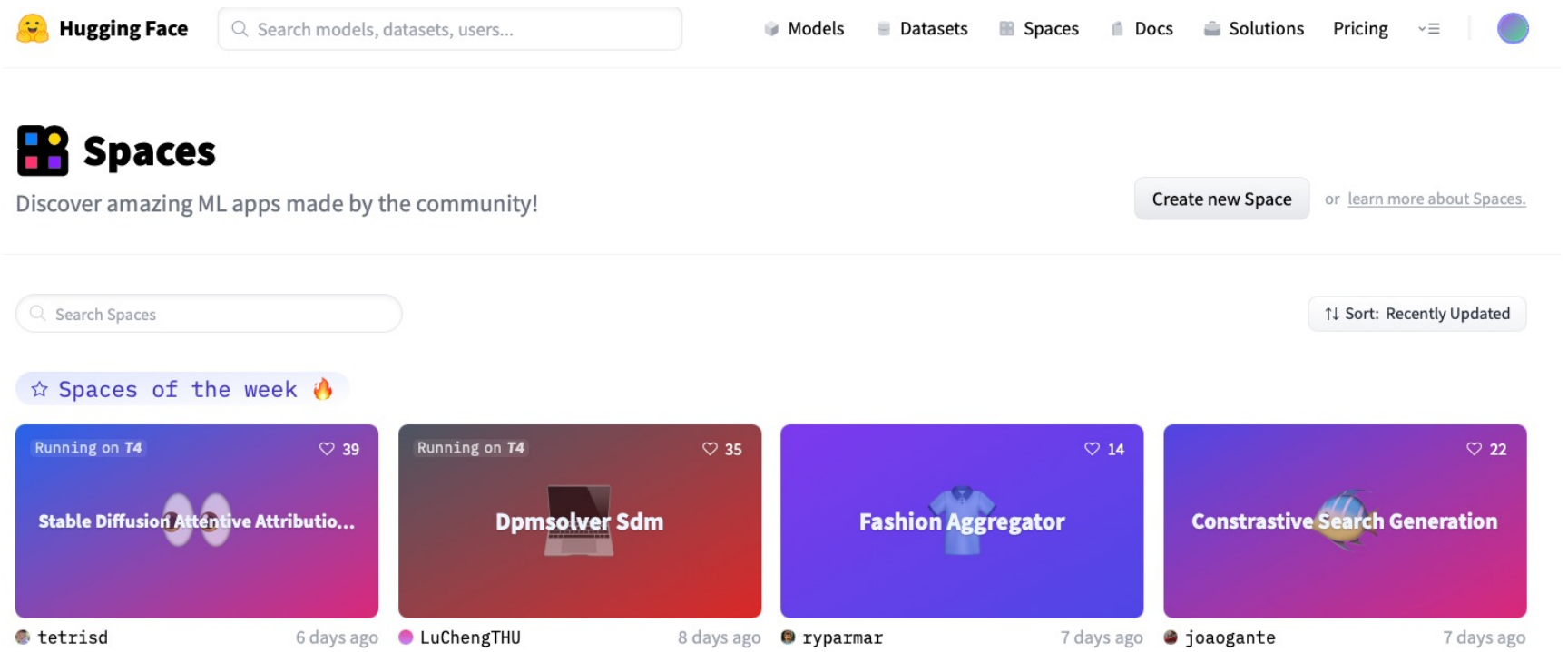
gradio.Video(…)

gradio.File(…)

gradio.JSON(…)

gradio.HTML(…)

Host the Gradio App on Spaces



1- Create a HuggingFace Account.
2- Join UVA-MSBA Organization by clicking on: https://huggingface.co/organizations/UVA-MSBA/share/rheZDkEGUpIgPbkXCgUXgxZJtwxhaCRHqc

Create a new Space

A Space is a special kind of repository that hosts application code for Machine Learning demos
Those applications can be written using Python libraries like **Streamlit** or **Gradio**

Owner

UVA-MSBA

/

Space name

New space name

License

License

Select the Space SDK

You can chose between Streamlit, Gradio and Static for your Space. Or pick Docker to host any other app.

Streamlit

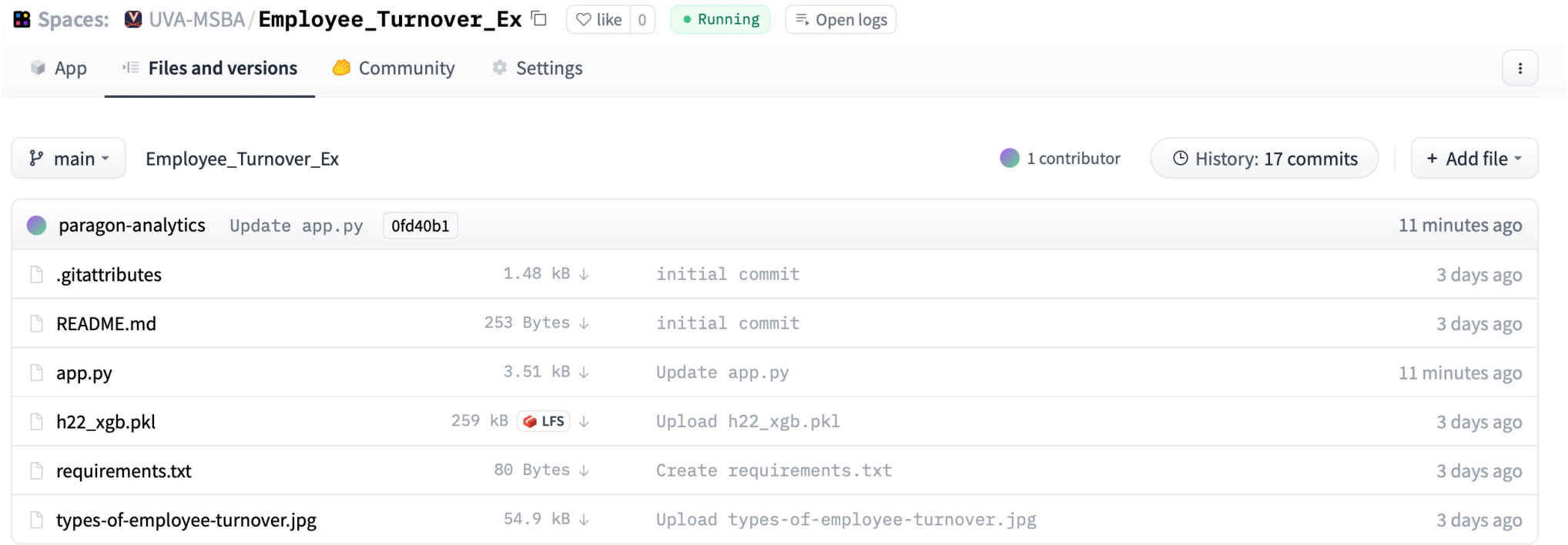
Gradio

NEW

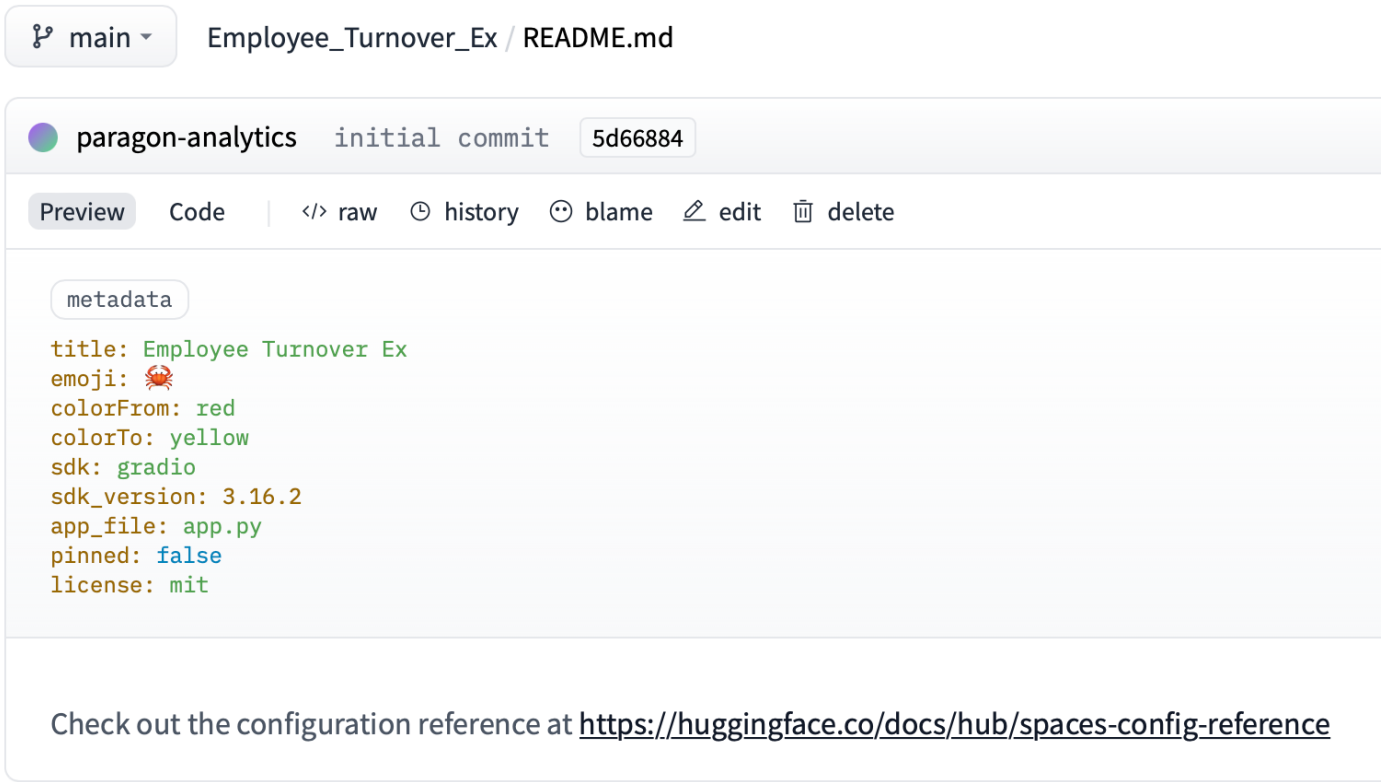
Docker

Static

Files


Spaces: UVA-MSBA / Employee_Turnover_Ex
like 0
Running
Open logs
App
Files and versions
Community
Settings
main
Employee_Turnover_Ex
1 contributor
History: 17 commits
+ Add file
paragon-analytics Update app.py 0fd40b1 11 minutes ago
.gitattributes 1.48 kB initial commit 3 days ago
README.md 253 Bytes initial commit 3 days ago
app.py 3.51 kB Update app.py 11 minutes ago
h22_xgb.pkl 259 kB LFS Upload h22_xgb.pkl 3 days ago
requirements.txt 80 Bytes Create requirements.txt 3 days ago
types-of-employee-turnover.jpg 54.9 kB Upload types-of-employee-turnover.jpg 3 days ago

README


main
Employee_Turnover_Ex / README.md
paragon-analytics
initial commit
5d66884
Preview
Code
raw
history
blame
edit
delete
metadata
title: Employee Turnover Ex
emoji:
colorFrom: red
colorTo: yellow
sdk: gradio
sdk_version: 3.16.2
app_file: app.py
pinned: false
license: mit
Check out the configuration reference at https://huggingface.co/docs/hub/spaces-config-reference

requirements.txt

main Employee_Turnover_Ex / requirements.txt

paragon-analytics Create requirements.txt b076948

raw history blame edit delete

```
1  gradio==3.1.3
2  Pillow
3  yake
4  pandas
5  sklearn
6  shap
7  xgboost
8  matplotlib
9  numpy
10 streamlit
```

app.py

raw history blame edit delete

```python
import pickle
import pandas as pd
import shap
from shap.plots._force_matplotlib import draw_additive_plot
import gradio as gr
import numpy as np
import matplotlib.pyplot as plt

# load the model from disk
loaded_model = pickle.load(open("h22_xgb.pkl", 'rb'))

# Setup SHAP
explainer = shap.Explainer(loaded_model) # PLEASE DO NOT CHANGE THIS.

```

model

Shap Explainer

app.py

Inputs

```
# Create the main function for server
def main_func(ValueDiversity,AdequateResources,Voice,GrowthAdvancement,Workload,WorkLifeBalance):
    new_row = pd.DataFrame.from_dict({'ValueDiversity':ValueDiversity,'AdequateResources':AdequateResources,
              'Voice':Voice,'GrowthAdvancement':GrowthAdvancement,'Workload':Workload,
              'WorkLifeBalance':WorkLifeBalance}, orient = 'index').transpose()

    prob = loaded_model.predict_proba(new_row)

    shap_values = explainer(new_row)
    # plot = shap.force_plot(shap_values[0], matplotlib=True, figsize=(30,30), show=False)
    # plot = shap.plots.waterfall(shap_values[0], max_display=6, show=False)
    plot = shap.plots.bar(shap_values[0], max_display=6, order=shap.Explanation.abs, show_data='auto', show=False)

    plt.tight_layout()
    local_plot = plt.gcf()
    plt.close()

    return {"Leave": float(prob[0][0]), "Stay": 1-float(prob[0][0])}, local_plot
```

Outputs

app.py

```
34  # Create the UI
35  title = "**Employee Turnover Predictor & Interpreter** 🪐"
36  description1 = """
37  This app takes six inputs about employees' satisfaction with different aspects of their work (such as work-life balance, ...) and predicts wh
38  """
39
40  description2 = """
41  To use the app, click on one of the examples, or adjust the values of the six employee satisfaction factors, and click on Analyze. ✨
42  """
```

app.py

```
with gr.Blocks(title=title) as demo:
    gr.Markdown(f"## {title}")
    # gr.Markdown("""![marketing](types-of-employee-turnover.jpg)""")
    gr.Markdown(description1)
    gr.Markdown("""---""")
    gr.Markdown(description2)
    gr.Markdown("""---""")
    # with gr.Row():
        # with gr.Column():
    ValueDiversity = gr.Slider(label="ValueDiversity Score", minimum=1, maximum=5, value=4, step=1)
    AdequateResources = gr.Slider(label="AdequateResources Score", minimum=1, maximum=5, value=4, step=1)
    Voice = gr.Slider(label="Voice Score", minimum=1, maximum=5, value=4, step=1)
    GrowthAdvancement = gr.Slider(label="GrowthAdvancement Score", minimum=1, maximum=5, value=4, step=1)
    Workload = gr.Slider(label="Workload Score", minimum=1, maximum=5, value=4, step=1)
    WorkLifeBalance = gr.Slider(label="WorkLifeBalance Score", minimum=1, maximum=5, value=4, step=1)
    submit_btn = gr.Button("Analyze")
        # with gr.Column(visible=True) as output_col:
    label = gr.Label(label = "Predicted Label")
    local_plot = gr.Plot(label = 'Shap:')

    submit_btn.click(
        main_func,
        [ValueDiversity,AdequateResources,Voice,GrowthAdvancement,Workload,WorkLifeBalance],
        [label,local_plot], api_name="Employee_Turnover"
    )
```

app.py

```
with gr.Blocks(title=title) as demo:
    gr.Markdown(f"## {title}")
    # gr.Markdown("""![marketing](types-of-employee-turnover.jpg)""")
    gr.Markdown(description1)
    gr.Markdown("""---""")
    gr.Markdown(description2)
    gr.Markdown("""---""")
    # with gr.Row():
        # with gr.Column():
    ValueDiversity = gr.Slider(label="ValueDiversity Score", minimum=1, maximum=5, value=4, step=1)
    AdequateResources = gr.Slider(label="AdequateResources Score", minimum=1, maximum=5, value=4, step=1)
    Voice = gr.Slider(label="Voice Score", minimum=1, maximum=5, value=4, step=1)
    GrowthAdvancement = gr.Slider(label="GrowthAdvancement Score", minimum=1, maximum=5, value=4, step=1)
    Workload = gr.Slider(label="Workload Score", minimum=1, maximum=5, value=4, step=1)
    WorkLifeBalance = gr.Slider(label="WorkLifeBalance Score", minimum=1, maximum=5, value=4, step=1)
    submit_btn = gr.Button("Analyze")
        # with gr.Column(visible=True) as output_col:
    label = gr.Label(label = "Predicted Label")
    local_plot = gr.Plot(label = 'Shap:')

    submit_btn.click(
        main_func,
        [ValueDiversity,AdequateResources,Voice,GrowthAdvancement,Workload,WorkLifeBalance],
        [label,local_plot], api_name="Employee_Turnover"
    )
```

app.py

```
    gr.Markdown("### Click on any of the examples below to see how it works:")
    gr.Examples([[4,4,4,4,5,5], [5,4,5,4,4,4]],
                [ValueDiversity,AdequateResources,Voice,GrowthAdvancement,Workload,WorkLifeBalance],
                [label,local_plot], main_func, cache_examples=True)

demo.launch()
```

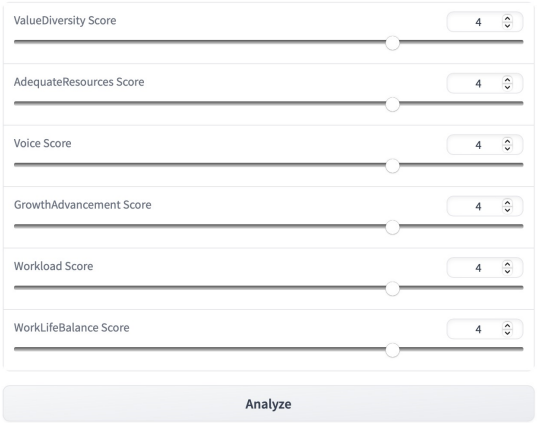
Questions to Answer with This App

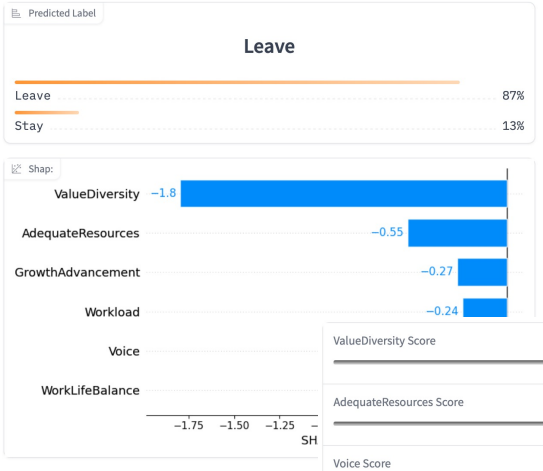
1. To what extent one unit improvement in ValueDiversity improves IntentionToStay?

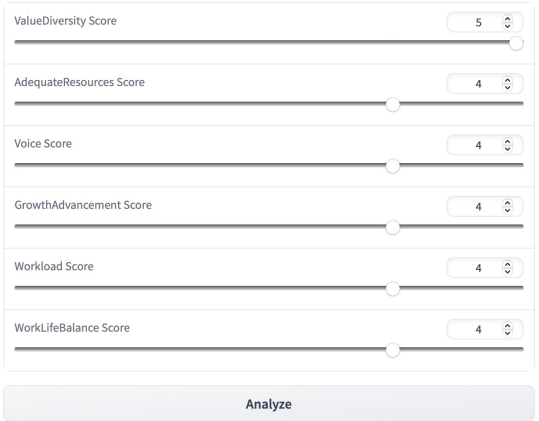
2. How much should we improve ValueDiversity so that IntentionToStay increases to over 50%?

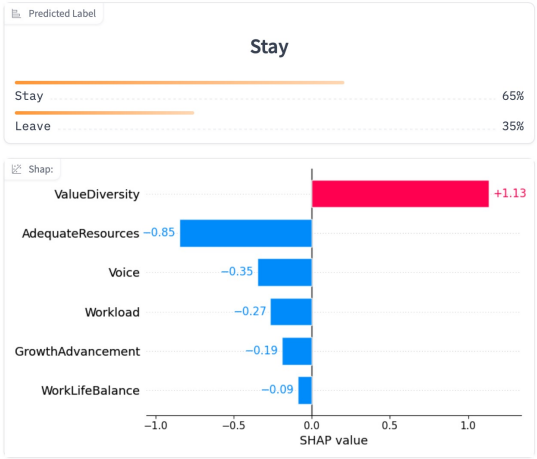
3. If we can only improve one factor, which factor should we improve?

To what extent one unit improvement in ValueDiversity improves IntentionToStay?

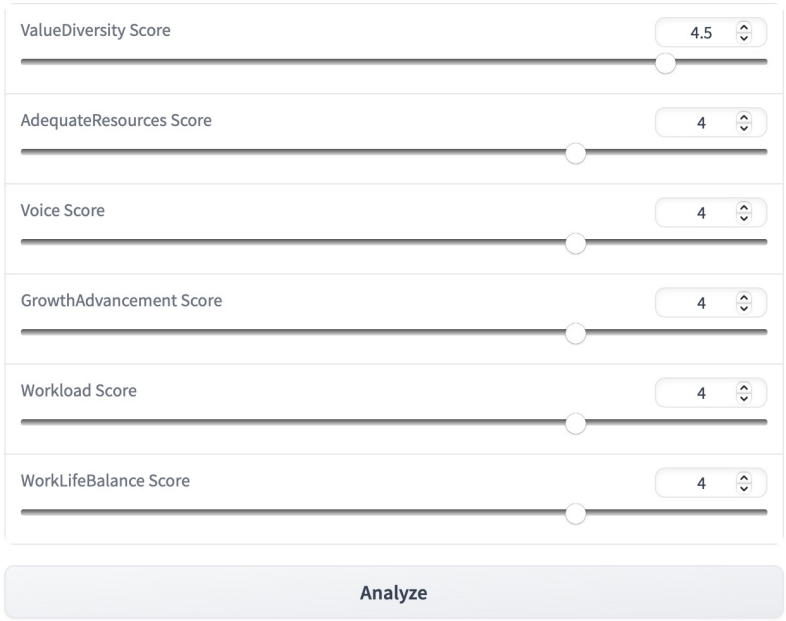


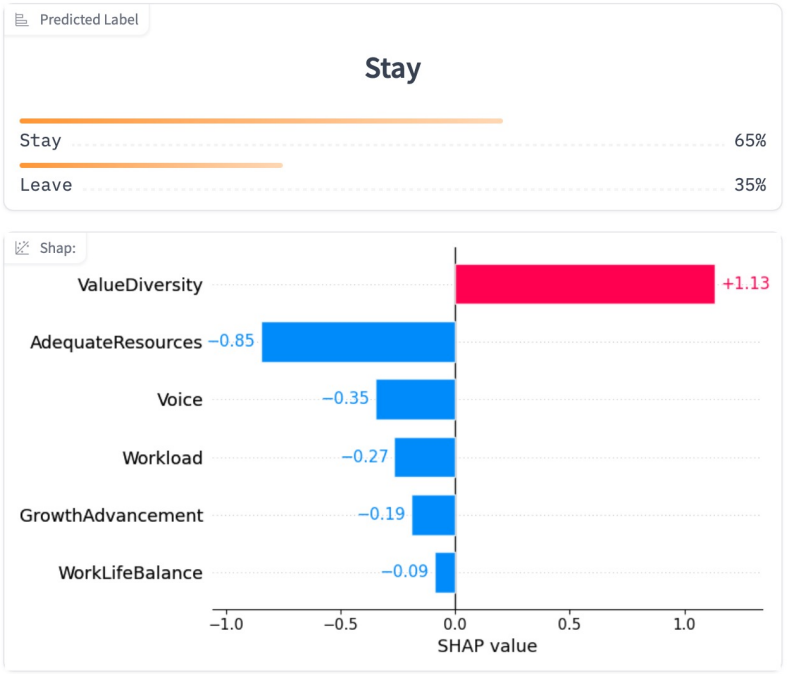




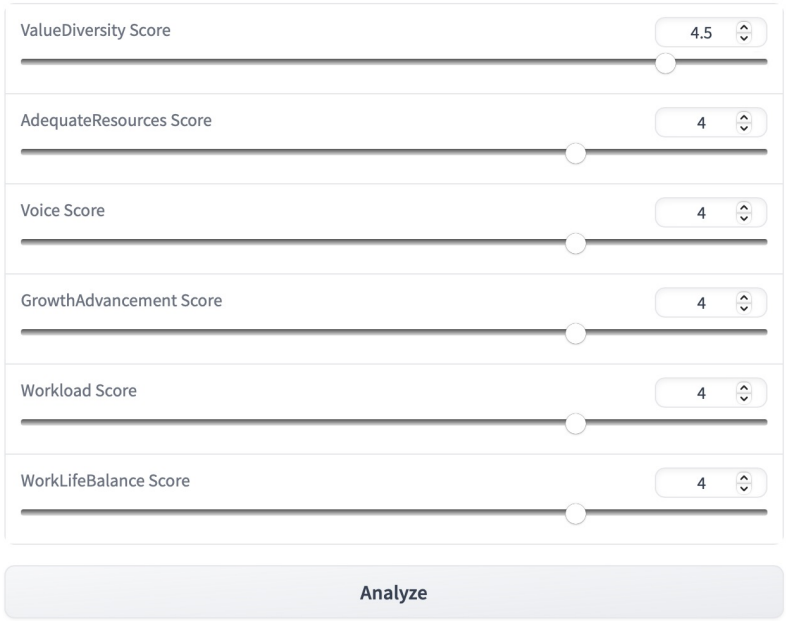


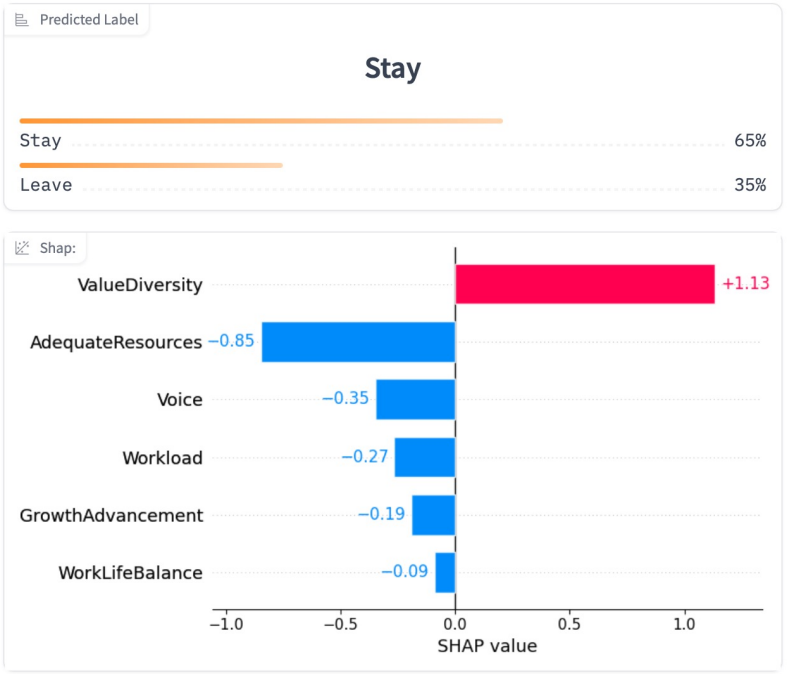
How much should we improve ValueDiversity so that IntentionToStay increases to over 50%?



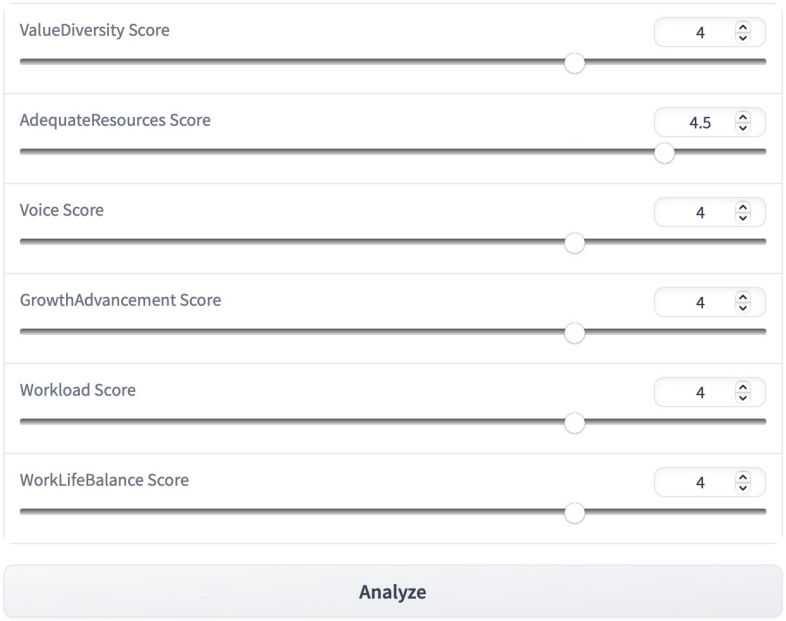


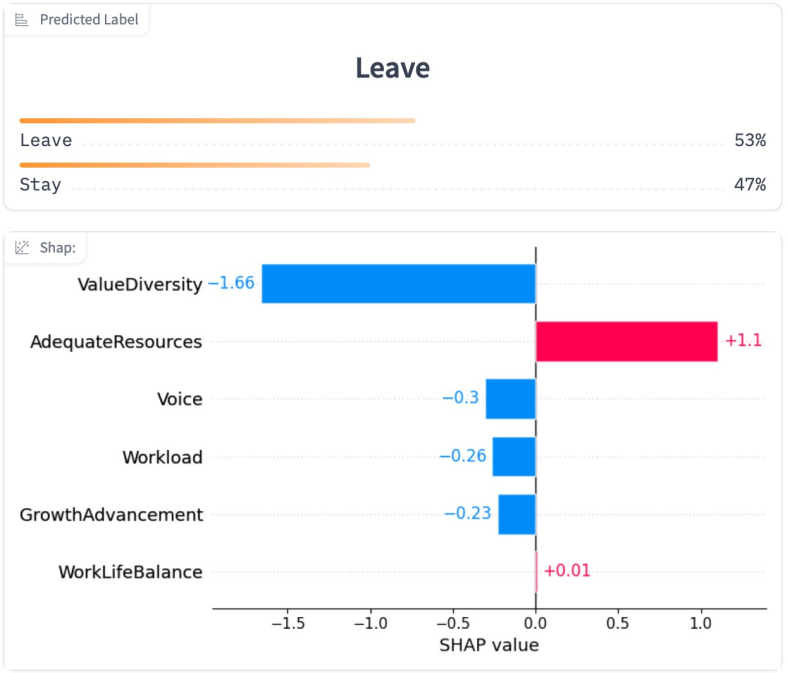
If we can only improve one factor, which factor should we improve? (ValueDiversity)



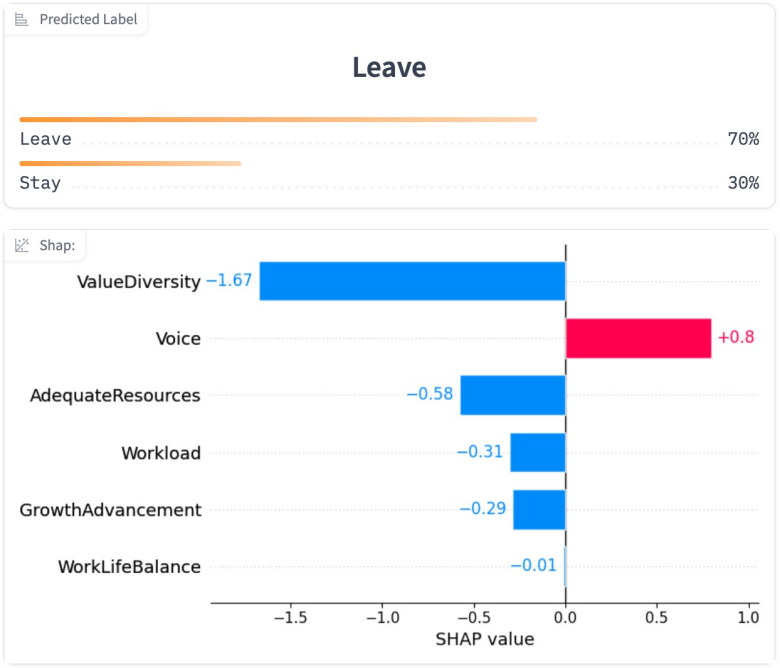


If we can only improve one factor, which factor should we improve? (AdequateResources)

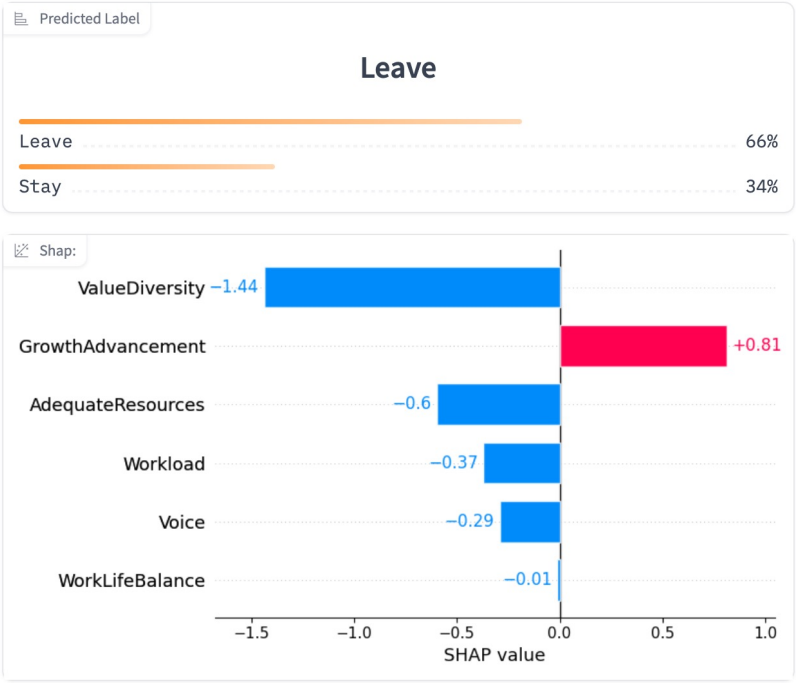




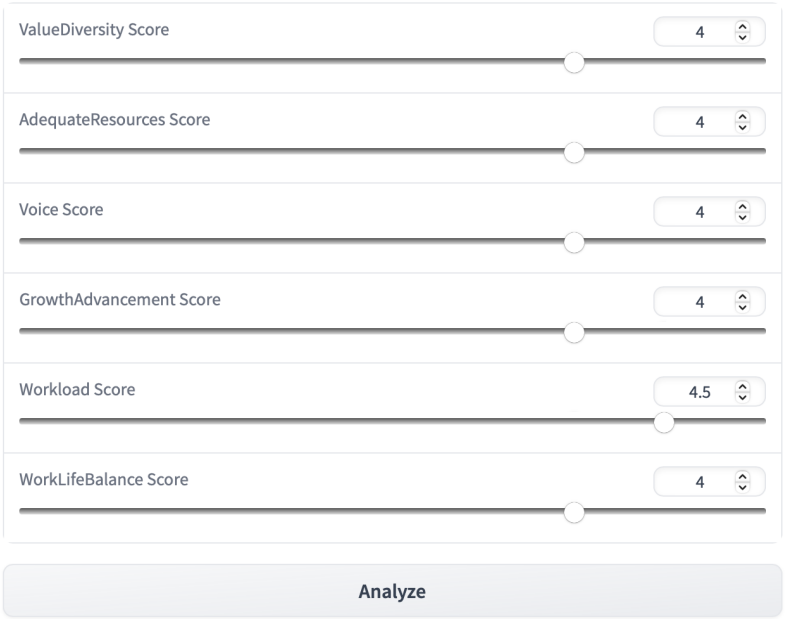
If we can only improve one factor, which factor should we improve? (Voice)

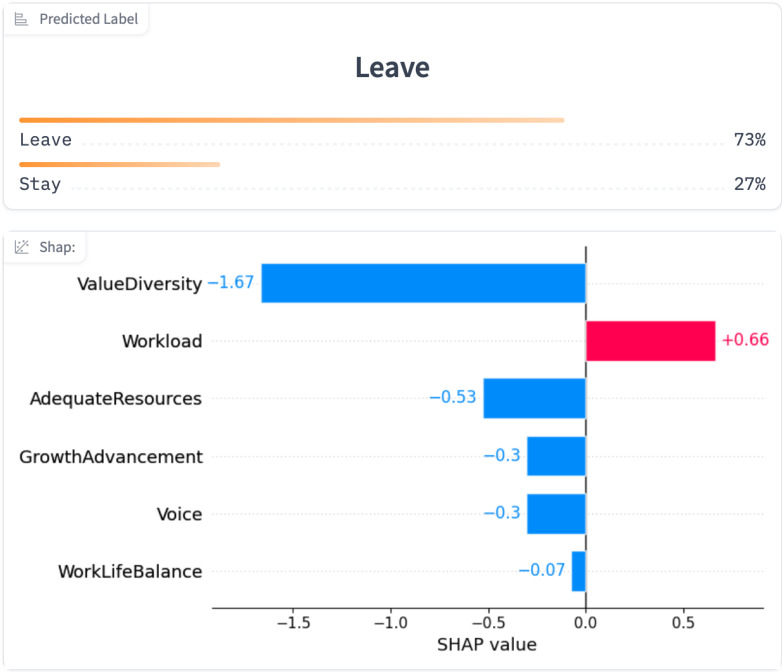


If we can only improve one factor, which factor should we improve? (GrowthAdvancement)

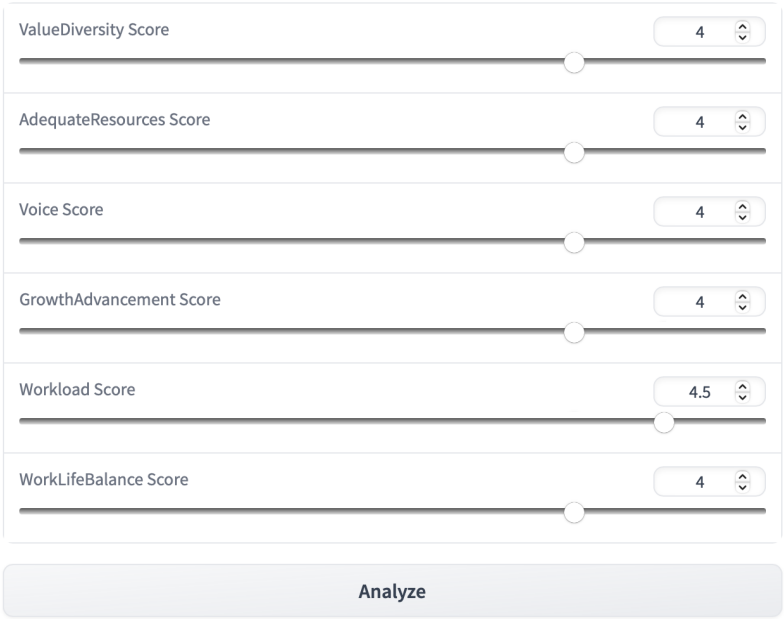


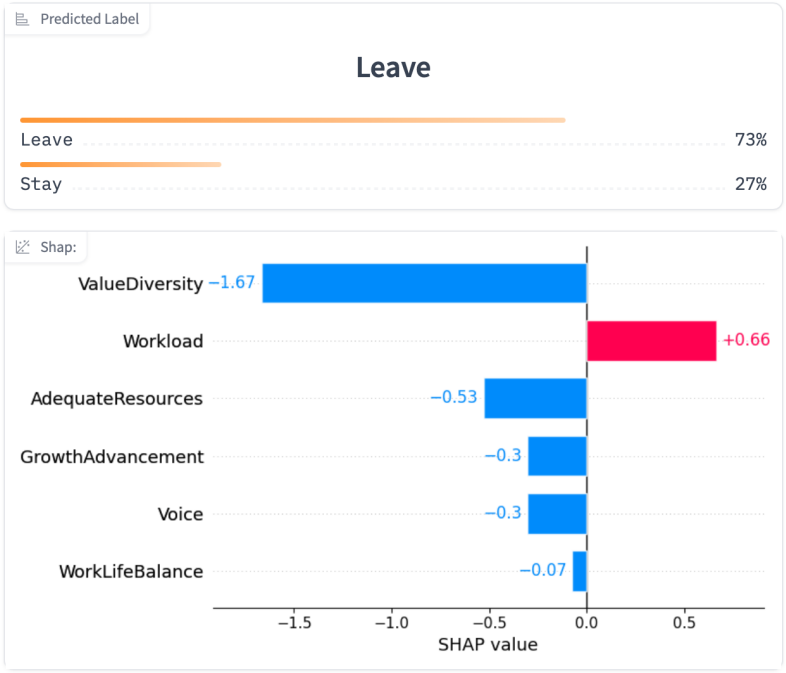
If we can only improve one factor, which factor should we improve? (Workload)



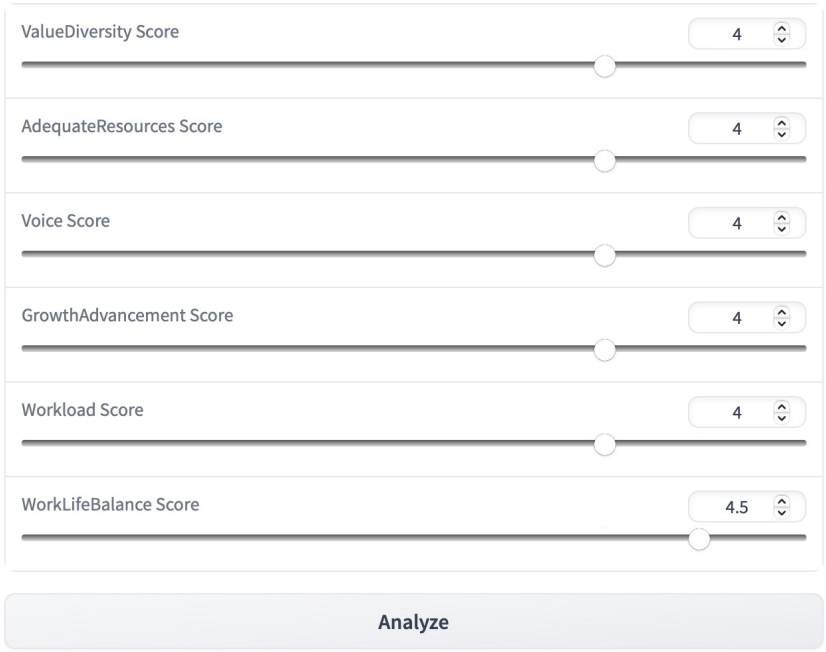


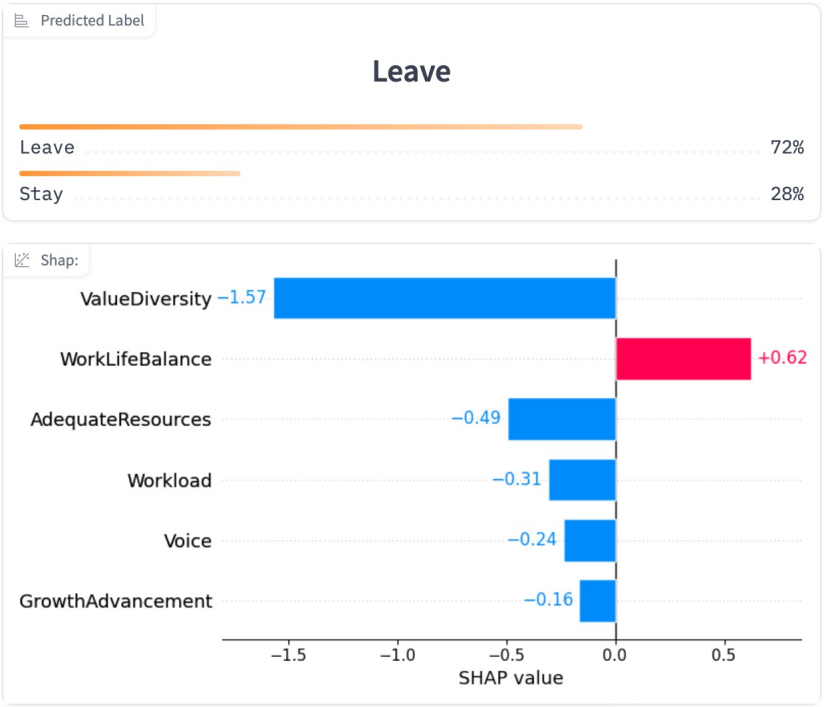
If we can only improve one factor, which factor should we improve? (Workload)





If we can only improve one factor, which factor should we improve? (WorkLifeBalance)





Demos and Tutorials

- https://huggingface.co/spaces/UVA-MSBA/Employee_Turnover_Ex
- https://huggingface.co/spaces/gradio/xgboost-income-prediction-with-explainability
- https://www.youtube.com/watch?v=RiCQzBluTxU
- https://www.youtube.com/watch?v=eE7CamOE-PA
- https://huggingface.co/spaces/paragon-analytics/ResText
- https://huggingface.co/spaces/paragon-analytics/Persuade

?


VIRGINIA