Stock Recommender App Deployment Guide

Using Claude AI, GitHub Actions, and Koyeb

1. Overview and Architecture

This guide will help you deploy a stock recommender app that runs on autopilot. The system consists of three main components:

1. **Python Script** - Downloads data from Rice stock database, runs your ML model, ranks stocks, saves to Excel
2. **Streamlit App** - Reads Excel file, displays Buy/Hold/Sell recommendations based on quartiles
3. **GitHub Actions** - Runs the Python script daily, commits results to your repo
4. **Koyeb Hosting** - Hosts your Streamlit app, auto-deploys when GitHub updates

How It Works

- Every day at a scheduled time, GitHub Actions runs your Python script
- Script downloads fresh data, runs predictions, saves ranked_predictions.xlsx
- GitHub automatically commits and pushes the updated Excel file
- Koyeb detects the new commit and redeploys your Streamlit app
- Users access your app anytime and see the latest predictions

2. Core Python Script Template

This script will download data, run your ML model, and save predictions. You'll customize this based on your specific data needs and model.

Step 1: Use Claude to Generate SQL Query

Before creating your script, use Claude with the **rice-data-query skill** to generate the SQL query for your specific data needs.

Example prompt to Claude:

```
"Get monthly prices and market cap for all stocks from 2020 to present"
```

Claude will provide you with the SQL query. Copy this query - you'll insert it into your Python script.

Step 2: Create predict_and_rank.py

Create a file called `predict_and_rank.py` with the following structure:

```
import pandas as pd
```

```
import duckdb
import joblib
from datetime import datetime

# 1. Download data from Rice database
sql_query = '''  # PASTE YOUR SQL FROM CLAUDE HERE
SELECT ticker, date, close, marketcap
FROM stocks
WHERE date >= '2020-01-01'
'''

con = duckdb.connect()
df = con.execute(sql_query).df()
con.close()

# 2. Load your trained ML model
model = joblib.load('my_model.pkl')  # Your saved model

# 3. Prepare features (customize for your model)
# Example: calculate returns, momentum, etc.
df['return_1m'] = df.groupby('ticker')['close'].pct_change()

# 4. Run predictions
features = df[['return_1m', 'marketcap']]  # Your features
df['predicted_return'] = model.predict(features)

# 5. Get most recent predictions and rank
latest = df.sort_values('date').groupby('ticker').tail(1)
ranked = latest.sort_values('predicted_return', ascending=False)

# 6. Save to Excel
ranked[['ticker', 'predicted_return']].to_excel(
    'ranked_predictions.xlsx', index=False
)
```

Step 3: Customize for Your Model

Replace the placeholders with your specific:

- SQL query (from Claude rice-data-query skill)
- Model file name and path
- Feature engineering code

- Feature column names

3. Streamlit App

The Streamlit app reads your Excel file and displays recommendations to users.

Create app.py

Create a file called `app.py` with the following code:

```
import streamlit as st
import pandas as pd
from datetime import datetime

st.title('Stock Recommendations')
st.write('Based on ML predictions')

# Load predictions
df = pd.read_excel('ranked_predictions.xlsx')

# Calculate quartiles
q1 = df['predicted_return'].quantile(0.25)
q3 = df['predicted_return'].quantile(0.75)

# Assign recommendations
df['recommendation'] = 'HOLD'
df.loc[df['predicted_return'] >= q3, 'recommendation'] = 'BUY'
df.loc[df['predicted_return'] <= q1, 'recommendation'] = 'SELL'

# Display summary
st.header('Recommendations Summary')
col1, col2, col3 = st.columns(3)
with col1:
    st.metric('BUY', (df['recommendation']=='BUY').sum())
with col2:
    st.metric('HOLD', (df['recommendation']=='HOLD').sum())
with col3:
    st.metric('SELL', (df['recommendation']=='SELL').sum())

# Show top BUY recommendations
st.header('Top BUY Recommendations')
```

```
buy_stocks = df[df['recommendation']=='BUY'].head(10)
st.dataframe(buy_stocks)

# Optional: show full ranked list
if st.checkbox('Show full ranked list'):
    st.dataframe(df)
```

Create requirements.txt

Create a file called `requirements.txt` listing all Python packages:

```
streamlit
pandas
openpyxl
duckdb
joblib
scikit-learn
```

4. Git and GitHub Setup

Step 1: Install Git

1. Go to https://git-scm.com/downloads
2. Download and install Git for your operating system
3. Open terminal/command prompt and verify installation:

```
git --version
```

Step 2: Create GitHub Account

4. Go to https://github.com
5. Click 'Sign up' and create a free account
6. Verify your email address

Step 3: Create Repository

7. Log into GitHub and click the '+' icon in top right
8. Select 'New repository'
9. Name it 'stock-recommender' (or your preferred name)
10. Set to Public (required for free Koyeb hosting)
11. Check 'Add a README file'
12. Click 'Create repository'

Step 4: Upload Your Files

You have two options for uploading files:

Option A: Web Interface (Easier for Beginners)

13. In your repository, click 'Add file' > 'Upload files'
14. Drag and drop all your files:
    - `predict_and_rank.py`
    - `app.py`
    - `requirements.txt`
    - Your trained model file (e.g., `my_model.pkl`)
15. Click 'Commit changes'

Option B: Command Line

In your project folder, run these commands:

```
git init

git add .

git commit -m "Initial commit"

git remote add origin https://github.com/YOUR-USERNAME/stock-recommender.git

git push -u origin main
```

5. GitHub Actions Workflow

GitHub Actions will run your prediction script daily and automatically commit the results.

Step 1: Create Workflow File

16. In your repository, create folder structure: `.github/workflows/`
17. Create file `daily_predictions.yml` in that folder
18. Copy this content:

```
name: Daily Stock Predictions


on:
  schedule:
    - cron: '0 10 * * *'  # 10 AM UTC daily
  workflow_dispatch:  # Allows manual trigger


jobs:
  predict:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3


      - name: Set up Python
```

```
      uses: actions/setup-python@v4
      with:
        python-version: '3.11'

    - name: Install dependencies
      run: pip install -r requirements.txt

    - name: Run predictions
      env:
        RICE_TOKEN: ${{ secrets.RICE_TOKEN }}
      run: python predict_and_rank.py

    - name: Commit results
      run: |
        git config --local user.email 'action@github.com'
        git config --local user.name 'GitHub Action'
        git add ranked_predictions.xlsx
        git commit -m 'Update predictions' || exit 0
        git push
```

Step 2: Configure Secrets

If your script needs the Rice database token:

19. Go to your repository Settings > Secrets and variables > Actions
20. Click 'New repository secret'
21. Name: `RICE_TOKEN`
22. Value: Your Rice database access token
23. Click 'Add secret'

Step 3: Test the Workflow

24. Go to Actions tab in your repository
25. Select 'Daily Stock Predictions' workflow
26. Click 'Run workflow' to test manually
27. Check that the workflow completes successfully
28. Verify that `ranked_predictions.xlsx` was updated in your repo

6. Koyeb Deployment

Koyeb will host your Streamlit app and automatically redeploy when GitHub updates.

Step 1: Create Koyeb Account

29. Go to https://www.koyeb.com

30. Click 'Sign up' and create a free account
31. You can sign up with your GitHub account (recommended)
32. Verify your email

Step 2: Create New App

33. In Koyeb dashboard, click 'Create App'
34. Select 'GitHub' as deployment method
35. Authorize Koyeb to access your GitHub account
36. Select your 'stock-recommender' repository

Step 3: Configure Deployment

Builder:
- Select 'Buildpack'

Build command:

```
pip install -r requirements.txt
```

Run command:

```
streamlit run app.py --server.port=8000 --server.address=0.0.0.0
```

Port: 8000
Instance type: Free (Nano)

Step 4: Deploy

37. Click 'Deploy'
38. Wait for deployment to complete (usually 2-5 minutes)
39. Koyeb will provide a public URL for your app
40. Click the URL to view your live app

Step 5: Enable Auto-Deploy

Koyeb automatically watches your GitHub repository. Every time GitHub Actions commits new predictions, Koyeb will:
- Detect the new commit
- Pull the latest code and data
- Redeploy your app with updated predictions

No additional configuration needed - this is automatic!

7. Verification and Testing

Verify Everything Works

41. **GitHub Actions runs successfully** - Check Actions tab
42. **Excel file is updated** - Check repository files
43. **Koyeb redeploys** - Check Koyeb dashboard
44. **App shows recommendations** - Visit your app URL

Troubleshooting

GitHub Actions fails:

- Check the Actions tab for error logs
- Verify RICE_TOKEN secret is set correctly
- Ensure all dependencies are in requirements.txt

Koyeb deployment fails:

- Check deployment logs in Koyeb dashboard
- Verify run command is correct
- Ensure port is set to 8000

App shows errors:

- Verify ranked_predictions.xlsx exists in repository
- Check that Excel file has correct column names
- Test app.py locally with: streamlit run app.py

Congratulations!

Your stock recommender app is now live and updates automatically every day. Users can visit your Koyeb URL anytime to see the latest recommendations.