Investor-Startup Matching: A Data Science approach at Parallel18

Introduction:

As a ventures Intern in Parallel18, they gave me the opportunity to apply my skill set in a business manner with the objective of optimizing a function. Parallel18 is a leading startup accelerator program that fosters the growth of innovative companies in Puerto Rico. Matching startups with ideal investors can be a critical challenge in the venture capital industry. This project aims to address this problem using a Data Science approach.

Data Acquisition:

The project utilized two datasets: the “Investor_data.csv” and the “P18_startup_Data.csv”, which were provided by the Ventures department.

Data Preprocessing:

1. **Handling Missing Values:** A few steps were taken in order to prepare the data for further steps, first the basics. For both datasets, we handled missing data. In the investor data set we used it for both the “Preferred Industry” and “Preferred stage” columns and were filled with “Not Specified”. In the startup dataset, for the “Industry” and “Description” columns were also filled with “Not Specified” for missing values.

```
investor_ds['Preferred Industry'].fillna('Not Specified', inplace=True)
investor_ds['Preferred Stage'].fillna('Not Specified', inplace=True)
startup_ds['Industry'].fillna('Not Specified', inplace=True)
startup_ds['Description'].fillna('Not Specified', inplace=True)
```

2. **Handling Duplicates**: Duplicates were removed from both the investor’s and startup’s datasets. Mainly the ID’s columns, respectfully.

```
investor_ds.drop_duplicates(subset='Investor ID', keep='first', inplace=True)
startup_ds.drop_duplicates(subset='Startup ID', keep='first', inplace=True)
```

3. **Feature Engineering:** Binary columns were created to represent the investor's preferred industry and stage. This was conducted by splitting the "Preferred Industry" and "Preferred Stage" columns and creating individual columns for each. The reason for this was to create a format that could be effectively used by the machine learning model. (Cosine Similarity)

```
for industry in industries:
    investor_ds[industry] = investor_ds['Preferred Industry'].apply(lambda x: 1 if industry in x.split(', ') else 0)
    startup_ds[industry] = startup_ds['Industry'].apply(lambda x: 1 if industry in x.split(',') else 0)
```

In the startup dataset another method was used in order to create unique columns for each of the stages that the startups are looking for.

```
stages = startup_ds['Stage'].str.split(',', expand=True).stack().unique()
stages = [stage.strip() for stage in stages if pd.notna(stage)]

for stage in stages:
    startup_ds[stage] = startup_ds['Stage'].str.contains(stage, case=False, na=False).astype(int)
```

4. **Data Transformation**: For the Investor dataset, the "Preferred Investment Amount" column was converted into a numeric format as an average of the stated ranges that are represented as "min to max".

```
def convert_to_numeric(amount_str):
    if isinstance(amount_str, str):
        amount_str = amount_str.strip()
        if 'to' in amount_str:
            start, end = amount_str.split('to')
            start_value = float(start.strip())
            end_value = float(end.strip())
            return (start_value + end_value) / 2
        return float(amount_str)

investor_ds['Preffered Investment Amount'] = investor_ds['Preffered Investment Amount'].apply(convert_to_numeric)
startup_ds['Amount Seeking'] = pd.to_numeric(startup_ds['Amount Seeking'], errors='coerce')
```

5. **Categorical Data Encoding:** For both the investor and startup datasets categorical features were encoded using a one-hot encoding approach in order to prepare them for the machine learning model. One-hot encoding is a technique used in Machine Learning that turns categorical data into numerical data.

```
encoder = OneHotEncoder(drop='first', sparse_output=False)
investor_encoded = encoder.fit_transform(investor_features[common_categorical_columns])
startup_encoded = encoder.transform(startup_features[common_categorical_columns])

investor_encoded_df = pd.DataFrame(investor_encoded, columns=encoder.get_feature_names_out(common_categorical_columns))
startup_encoded_df = pd.DataFrame(startup_encoded, columns=encoder.get_feature_names_out(common_categorical_columns))
```

6. **Numeric Feature Scaling:** The Scikit-learn library's feature "StandardScaler" was used in order to scale the numeric features in both the investor and startup datasets.

```
investor_numeric = investor_features.select_dtypes(include=[np.number])
startup_numeric = startup_features.select_dtypes(include=[np.number])

common_numeric_columns = investor_numeric.columns.intersection(startup_numeric.columns)
investor_numeric = investor_numeric[common_numeric_columns]
startup_numeric = startup_numeric[common_numeric_columns]

scaler = StandardScaler()
investor_numeric_scaled = scaler.fit_transform(investor_numeric)
startup_numeric_scaled = scaler.transform(startup_numeric)
```

The first two lines identify the numeric columns from the investor and startup feature sets. The next three lines find the columns that are common in both datasets, then filter both the datasets to include only these columns. This ensures that we're comparing the same types of numeric data for both the investors and startups.

Then the StandardScaler is applied to normalize the numeric data. This is crucial because we want to ensure that all features contribute equally to our similarity calculations.

Model Development:

For this project, a Cosine-Similarity approach is used to match investors with startups. Cosine similarity is a measure of similarity between two non-zero vectors in an n-dimensional space and for recommendation systems it is quite accurate.

1. **Investor and Startup Feature Representation:** The preprocessed investor and startup datasets are combined into two separate features matrices, investor_combined and startup_combined, respectively. This step prepares the data for the cosine similarity calculation.

```
investor_combined = np.hstack([investor_numeric_scaled, investor_encoded])
startup_combined = np.hstack([startup_numeric_scaled, startup_encoded])
```

2. **Cosine Similarity Calculation:** The cosine similarity between the new investor data and the startup data was computed, resulting in a similarity matrix.

```
similarity_matrix = cosine_similarity(new_investor_combined, startup_combined)
```

3. **Recommendation Generation:** The top 15 investors are selected with the highest similarity scores depending on the requirements of the startup.

```
similarity_df = pd.DataFrame(similarity_matrix, index=[0], columns=investor_ds['Investor ID'])
recommendations = similarity_df.iloc[0].nlargest(15)
```

Web Application Development

For in-house use, after the project was finished. I wanted to facilitate the usage for the whole team. I came across a Python package called Flask, which is a web framework that's used for web applications. The UI was created as a form asking the user for Startups information: Industry, Stage, Amount Seeking.

1. **Form Submission:** The web application provides a form where users can input their investor information, including Industry, Stage, amount seeking.

```
@app.route('/')
def form():
    return render_template('startup_form.html')  # HTML for UI
```

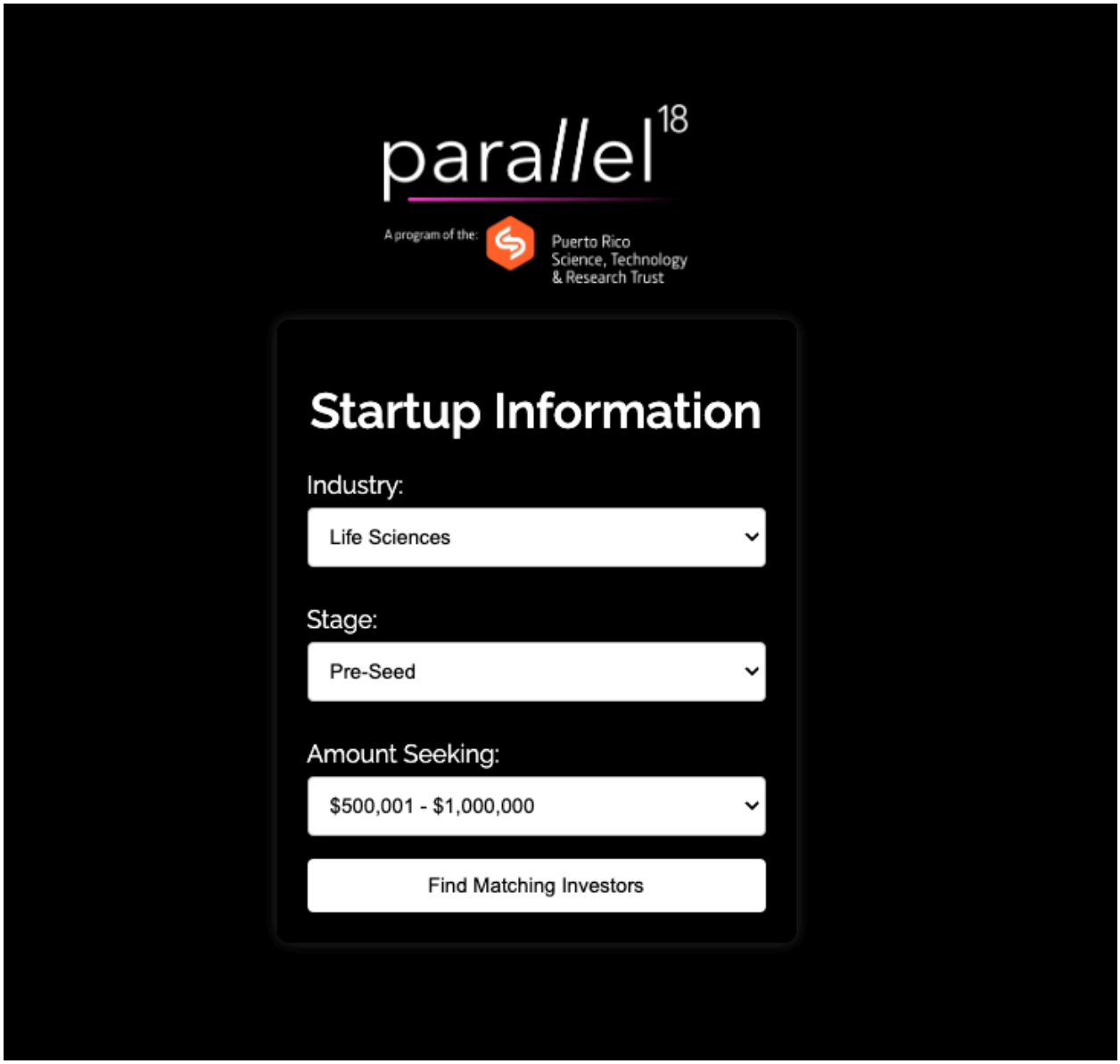
2. **Recommendation Generation:** When the form is submitted, the application processes the input data, generates the recommendations using the cosine similarity model, and displays the top 15 investors with their information and similarity scores.

```python
@app.route('/recommend', methods=['POST'])
def recommend():
    try:
        industry = request.form['Industry']
        stage = request.form['Stage']
        amount_seeking = request.form['Amount Seeking']

        # Handle the amount seeking range
        if amount_seeking == '5000000+':
            amount_value = 5000001
        else:
            amount_range = amount_seeking.split('-')
            amount_value = (float(amount_range[0]) + float(amount_range[1])) / 2

        # Create a new startup entry
        new_startup = pd.DataFrame({
            'Industry': [industry],
            'Stage': [stage],
            'Amount Seeking': [amount_value]
        })

        # Add industry columns
        for ind in industries:
            new_startup[ind] = 1 if ind in industry.split(', ') else 0

        # Add stage columns
        for stg in stages:
            new_startup[stg] = 1 if stg in stage else 0

        # Add 'All industries' column
        new_startup['All industries'] = 1 if 'All industries' in industry else 0

        # Ensure all columns from startup_features are present
        for col in startup_features.columns:
            if col not in new_startup.columns:
                new_startup[col] = 0
```

```python
        similarity_df = pd.DataFrame(similarity_matrix, index=[0], columns=investor_ds['Investor ID'])
        recommendations = similarity_df.iloc[0].nlargest(15)

        # Fetch investor details for the top recommendations
        top_investors = investor_ds[investor_ds['Investor ID'].isin(recommendations.index)]
        top_investors['Similarity Score'] = recommendations.values
        top_investors = top_investors[['Name', 'Email', 'Preferred Industry', 'Preferred Stage', 'Preffered Investment Amount', 'Similarity Score']]

        return top_investors.to_html(index=False)

        new_investor.to_csv('new_investors.csv', mode='a', header=False, index=False)

    except ValueError as e:
        return f"Error processing input: {str(e)}", 400
    except Exception as e:
        return f"An unexpected error occurred: {str(e)}", 500

if __name__ == '__main__':
    app.run(debug=True, port=5009)
```

UI preview:

Form:



Results:

Name	Email	Preferred Industry	Preferred Stage	Preffered Investment Amount	Similarity Score
Sergey Vasnetsov	svex16@gmail.com	Not Specified	Pre-seed, Seed, Series A, Series B, Series C, Series D, Series E	NaN	0.624641
Ramiro Armando Zegarra Rivera	ramiro@avpventures.com	Not Specified	Pre-seed, Seed	75000.0	0.492487
Todd Swanson	todd@tqswanson.com	All Industries	Pre-seed, Seed, Series A, Series B, Series C, Series D, Series E	1005000.0	0.364375
Juan Mercado	juan.mercado@500.co	All Industries	Pre-Seed, Seed	300000.0	0.364375
Stefano Camilo Arrigoni Samso	stefano.arrigoni@dualcapital.cl	All Industries	Pre-seed, Seed, Series A, Series B, Series C, Series D, Series E	300000.0	0.364375
Jose F. Rodriguez Orengo	jforengo139@gmail.com	Healthcare and Biotechnology, Life Sciences, Artificial Intelligence and Machine Learning, Gaming and Esports	Pre-Seed, Seed, Series A	30000.0	0.364375
Francisco Uriarte	francisco@connellycap.com	All industries	Pre-seed, Seed, Series A, Series B, Series C, Series D, Series E	30000.0	0.364375
Emmanuel Loubriel	eloubriel@cdvca.org	All industries	Pre-Seed, Seed, Series A	300000.0	0.364375
Orion Choy	orion@touchdownvc.com	All industries	Pre-Seed, Seed, Series A, Series B, Series C, Series D, Series E	NaN	0.364375
Scott Meier	scott@budsies.com	All industries	Pre-Seed, Seed, Series A, Series B	NaN	0.364375
DJ Lampitt	dj@ideationcolab.com	All industries	Pre-Seed, Seed	30000.0	0.364375
Cesar De La Cruz	cesar.delacruz@popular.com	Healthcare and Biotechnology, Life Sciences, Financial Technology, Education Technology, Agriculture Technology, Artificial Intelligence and Machine Learning	Seed, Series A, Series B, Series C, Series D, Series E	300000.0	0.364375
Laura Cantero	lcantero@guayacan.org	All industries	Pre-Seed, Seed	30000.0	0.364375
Rouven Heck	investments@heck.cc	All industries	Pre-Seed, Seed	55000.0	0.364375
Alberto Estrella	agestrella@estrellallc.com	All industries	Pre-Seed	30000.0	0.364375

Evaluation and Future Improvements:

The project has successfully demonstrated the integration of data science techniques into a functional web application for startup-investor matching. However, there are some areas for potential improvements:

1. **Expanding the Dataset**:
    - Incorporate additional data sources, such as industry trends, market dynamics, and startup-investor relationship histories.
2. **Incorporating Feedback and Preferences**:
    - Allow investors and startups to provide feedback on the recommendations and update the model accordingly.
3. **Exploring Alternative Machine Learning Techniques**:
    - Experiment with other machine learning algorithms, such as collaborative filtering or hybrid models.
4. **Enhancing the Web Application**:
    - Add features like user authentication, personalized dashboards, and the ability to save and manage favorite startups.

6. Conclusion

This data science project has successfully developed a recommendation system to match startups with investors, leveraging data preprocessing, feature engineering, and machine learning techniques. The integration of the model into a web application provides a practical solution for the venture capital industry. By addressing the key challenges and exploring future improvements, this project demonstrates the potential of data science to drive innovation and facilitate more effective connections between investors and startups.