MSDS 422- Final Report

Abstract

This project investigates the relationship between academic success and career outcomes using machine learning models to predict key career metrics such as starting salary. The chosen data frame consists of various academic and professional factors, including High School GPA, SAT scores, University GPA, internships, projects, certifications, soft skills, and networking scores from about 5,000 individuals. With the chosen dataset, our work applies and compares various ML models, such as Gradient Boosting, Logistic Regression, Random Forest, and Decision Tree in order to predict student success and career readiness. Furthermore, this analysis on these factors allows us to explore career outcomes such as job offers, salary, career satisfaction, and promotions.

Literature Review

A study by H. Adu-Twum (2024) highlights the use of advanced predictive analytics and machine learning models to address student dropouts in higher education. The study employs similar models, such as Logistic Regression, Random Forest, Decision Tree Classifier, Support Vector Machine (SVM), and Gradient Boosting, to predict the likelihood of students dropping out of higher education based upon variables such as demographics, socioeconomics, academics, and financials.

Another study by Guleria and Sood (2022) employs both machine learning and AI models, such as White Box (e.g., Decision Trees, Naive Bayes) and Black Box (e.g., Ensemble Models, SVM) ML models to analyze educational datasets for career guidance based on various student features. This study really highlights the importance of cross-referencing results on multiple models by analyzing common challenges in ML applications. This reinforces our focus on utilizing multiple model evaluations and performance metrics in our current project.

Findings and Conclusion

In this project, some key insights were observed. Our exploratory data analysis showed that certain variables like High School GPA, SAT scores, and University GPA showed a weak correlation with starting salary, which goes against many traditional assumptions. This is further backed by our four predictive models, where our results confirmed that these factors alone lack predictive power. On the other hand, factors such as internships, projects, certifications, and soft skills slightly did contribute to the model but did not significantly improve predictive accuracy, concluding that these combinations of factors were also insufficient to build a strong predictive model for salary outcomes. However, our models overall did perform well. These models provide pretty strong performance despite the non-diverse dataset in predicting student success based on the RMSE and R squared values. Linear Regression and Neural Networks (MLP) performed slightly worse, with higher RMSE values and negative R^2 scores. While these two models didn’t outperform Random Forest and Gradient Boosting, they still offered good insights and allowed us to compare overall performances effectively.

Lessons Learned and Recommendations

One of the most important lessons from this project was realizing the need for comprehensive and detailed datasets. Our model's weaknesses opened up a lot of room for learning- effectively suggesting that many datasets can lack characteristics in building a proper model- in our case industry experience, and personal networks, among others. These challenges open up room for improvement, and for further research it would be highly suggested to find or gather more detailed and diverse data, potentially including factors such as personal connections, market conditions, employer reputation, negotiation skills, and regional differences, etc..

References

Adu-Twum, Harold Tobias, Emmanuel Adu Sarfo, Evans Nartey, Adesola Adetunji, Adebowale Olufemi Ayannusi, and Thomas Andrew Walugembe. “Role of Advanced Data Analytics in Higher Education: Using Machine Learning Models to Predict Student Success.” *International Journal of Computer Applications Technology and Research* 13, no. 08 (2024): 54-61.

Guleria, Pratiyush, and Manu Sood. “Explainable AI and Machine Learning: Performance Evaluation and Explainability of Classifiers on Educational Data Mining Inspired Career Counseling.” *Education and Information Technologies* 28 (2022): 1081-1116.

Appendix:

Project: Using Machine Learning To Predict Career Success Based on Academic Success

Objective: Analyze factors influencing career success, including academic performance, internships, certifications, soft skills, and networking, to understand their impact on job offers, salary, career satisfaction, and promotions.

Problem Statement:

1. What academic metrics (High School GPA, SAT Score, University GPA) correlate most with career outcomes?
2. Do internships, projects, or certifications improve job offers or salaries?
3. How do soft skills and networking scores influence promotions and career satisfaction?
4. Are there disparities in outcomes by gender or field of study?
5. Can we predict salary or career satisfaction using other variables?

Key Variables:

Educational Background: High school GPA, SAT score, university ranking, university GPA, and field of study. Professional Experience: Internships completed, projects completed, certifications, soft skills score, networking score. Career Outcomes: Number of job offers, starting salary, career satisfaction, years to promotion, current job level, work-life balance, and entrepreneurship status.

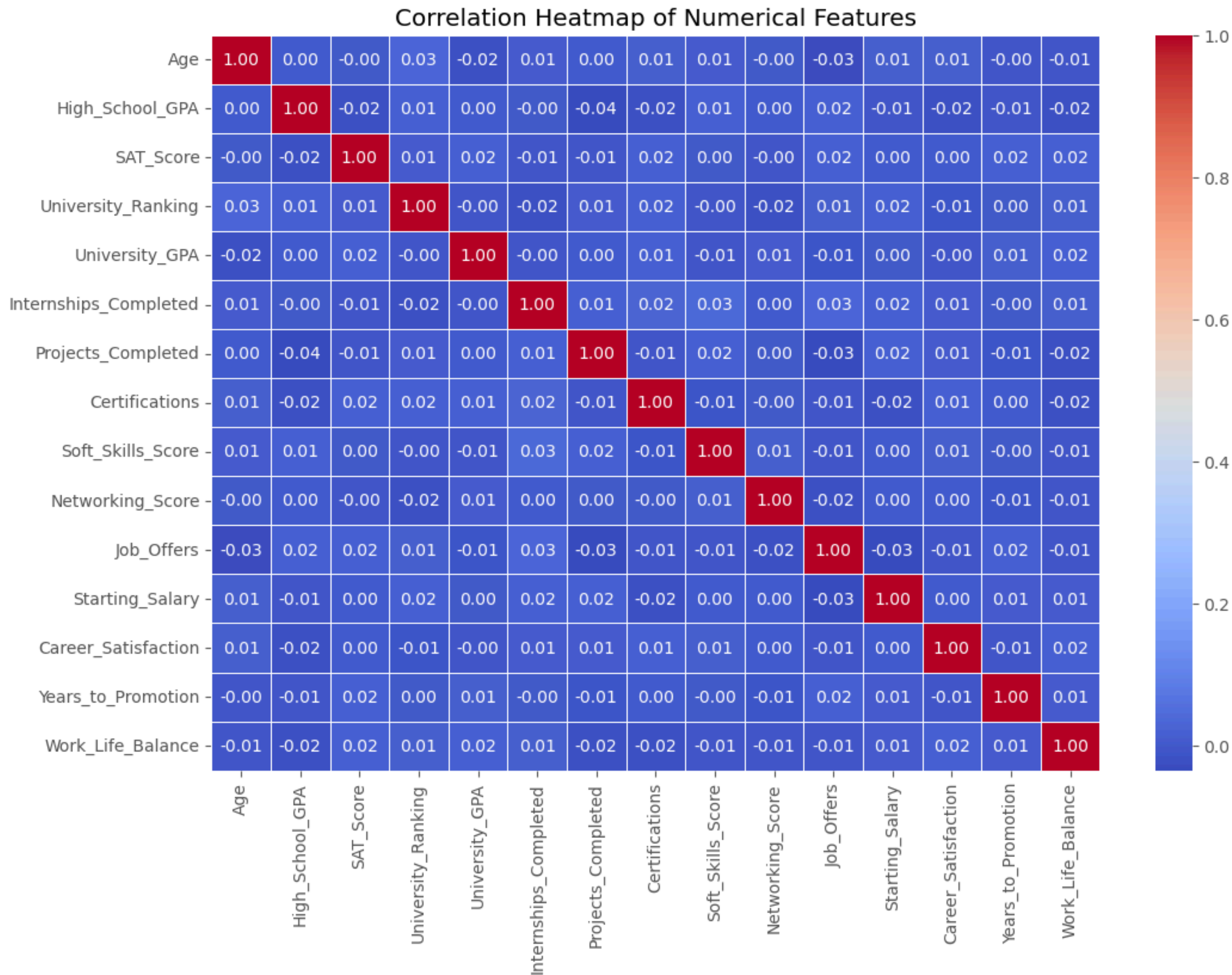
In [18]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


file_path = "education_career_success.csv"
df = pd.read_csv(file_path)

# Basic and Table Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 20 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Student_ID             5000 non-null   object 
 1   Age                    5000 non-null   int64  
 2   Gender                 5000 non-null   object 
 3   High_School_GPA        5000 non-null   float64
 4   SAT_Score              5000 non-null   int64  
 5   University_Ranking     5000 non-null   int64  
 6   University_GPA         5000 non-null   float64
 7   Field_of_Study         5000 non-null   object 
 8   Internships_Completed  5000 non-null   int64  
 9   Projects_Completed     5000 non-null   int64  
 10  Certifications         5000 non-null   int64  
 11  Soft_Skills_Score      5000 non-null   int64  
 12  Networking_Score       5000 non-null   int64  
 13  Job_Offers             5000 non-null   int64  
 14  Starting_Salary        5000 non-null   float64
 15  Career_Satisfaction    5000 non-null   int64  
 16  Years_to_Promotion     5000 non-null   int64  
 17  Current_Job_Level      5000 non-null   object 
 18  Work_Life_Balance      5000 non-null   int64  
 19  Entrepreneurship       5000 non-null   object 
dtypes: float64(3), int64(12), object(5)
memory usage: 781.4+ KB
```

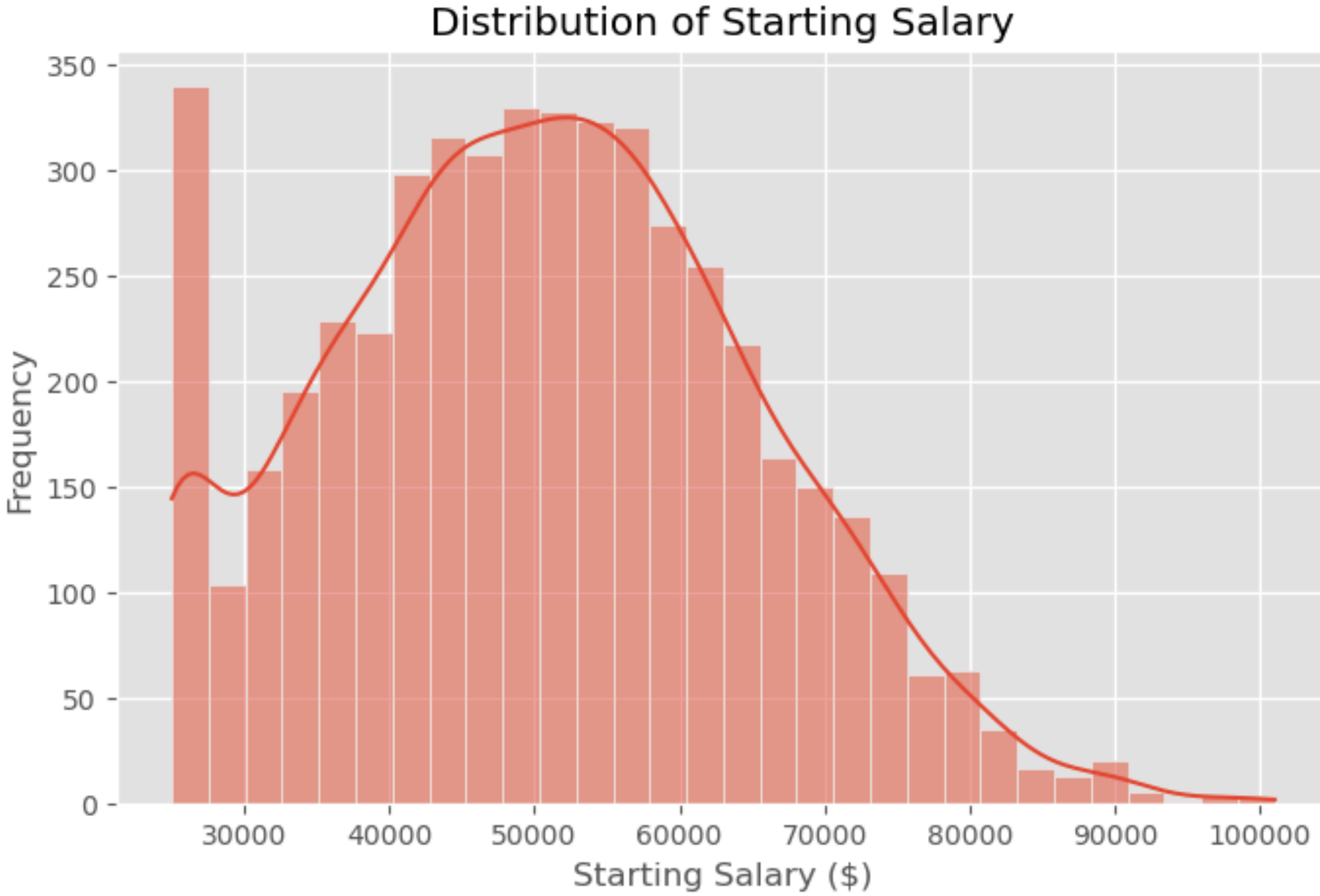
In [20]:

```
#Plot Style
plt.style.use("ggplot")

# Correlation heatmap for numerical variables
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, fmt=".2f", cmap="coolwarm", linewidt
plt.title("Correlation Heatmap of Numerical Features")
plt.show()
```



In [22]:

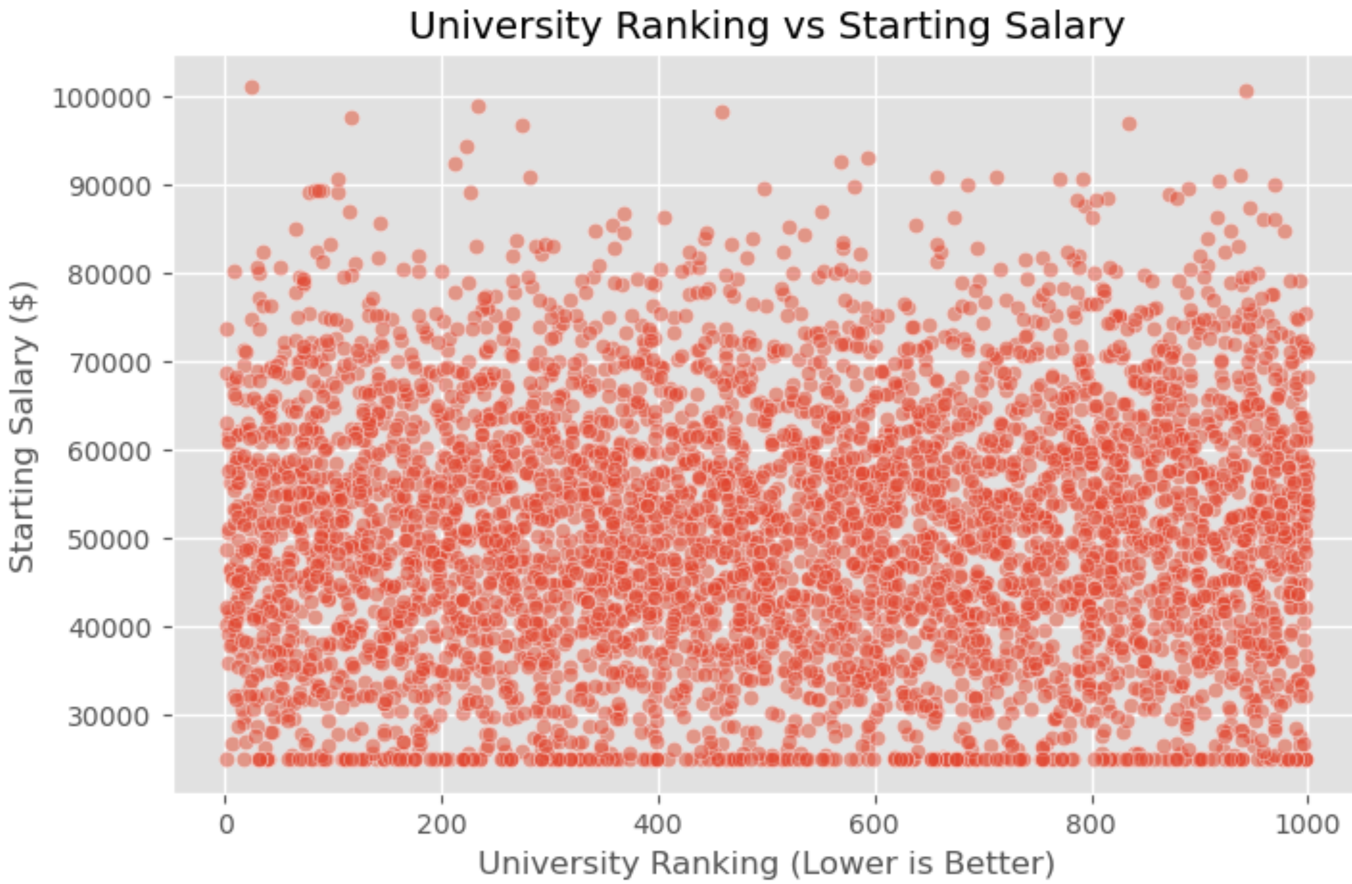
```
# Distribution of starting salary
plt.figure(figsize=(8, 5))
sns.histplot(df["Starting_Salary"], bins=30, kde=True)
plt.title("Distribution of Starting Salary")
plt.xlabel("Starting Salary ($)")
plt.ylabel("Frequency")
plt.show()
```



In [24]:
```
# Boxplot of career satisfaction by job level
plt.figure(figsize=(8, 5))
sns.boxplot(x="Current_Job_Level", y="Career_Satisfaction", data=df, order=["Entry", "Mi
plt.title("Career Satisfaction by Job Level")
plt.xlabel("Job Level")
plt.ylabel("Career Satisfaction Score")
plt.show()
```

In [26]:

```
# Relationship between university ranking and starting salary
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df["University_Ranking"], y=df["Starting_Salary"], alpha=0.5)
plt.title("University Ranking vs Starting Salary")
plt.xlabel("University Ranking (Lower is Better)")
plt.ylabel("Starting Salary ($)")
plt.show()
```



In [32]:

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
import numpy as np

df_cleaned = df.copy()

df_cleaned.dropna(inplace=True)  # Drop rows with missing values

#Encoding Categorical Variables
categorical_features = ["Gender", "Field_of_Study", "Current_Job_Level", "Entrepreneursh
label_encoders = {}

for col in categorical_features:
    le = LabelEncoder()
    df_cleaned[col] = le.fit_transform(df_cleaned[col])
    label_encoders[col] = le  # Store encoder for reference

#Feature Transformation & Scaling
scaler = StandardScaler()
numerical_features = ["High_School_GPA", "SAT_Score", "University_Ranking", "University_
                      "Internships_Completed", "Projects_Completed", "Certifications",
                      "Soft_Skills_Score", "Networking_Score", "Starting_Salary",
                      "Years_to_Promotion", "Work_Life_Balance"]

df_cleaned[numerical_features] = scaler.fit_transform(df_cleaned[numerical_features])

#Create New Features
df_cleaned["Work_Experience_Score"] = (
    df["Internships_Completed"] * 2 + df["Projects_Completed"] + df["Certifications"] *
)

# Feature Selection - Removing Unnecessary Columns
df_cleaned.drop(columns=["Student_ID"], inplace=True)  # Remove ID column

#Display cleaned set
print(df_cleaned.head())
```

Loading [MathJax]/extensions/Safe.js

```
   Age  Gender  High_School_GPA  SAT_Score  University_Ranking  \
0   24       1         1.012867  -0.993226           -0.733034   
1   21       2        -0.828640  -0.210778           -1.348089   
2   28       0         0.734903  -0.299357            0.723856   
3   25       1        -0.984994   1.196642           -1.148798   
4   22       1        -1.593038  -1.190068            0.325273   

   University_GPA  Field_of_Study  Internships_Completed  Projects_Completed  \
0        1.631925               0               0.722829            0.848418   
1        1.058998               4               1.433017            0.848418   
2       -0.677144               6               1.433017            1.196530   
3       -0.364638               2               0.722829            1.544642   
4       -0.937565               3               1.433017            0.500306   

   Certifications  Soft_Skills_Score  Networking_Score  Job_Offers  \
0       -0.300761           1.211558          0.863920           5   
1        0.286434           0.860789         -1.592393           4   
2       -0.887956          -1.594599          1.214822           0   
3       -0.887956           1.562328          0.162117           1   
4        0.873628           1.562328          1.214822           4   

   Starting_Salary  Career_Satisfaction  Years_to_Promotion  \
0        -1.612000                    4            1.399982   
1        -1.763792                    1           -1.422278   
2        -0.563255                    9           -0.011148   
3         0.471691                    7            1.399982   
4        -0.204474                    9            1.399982   

   Current_Job_Level  Work_Life_Balance  Entrepreneurship  \
0                  0           0.526371                 0   
1                  2           0.526371                 0   
2                  0           0.526371                 0   
3                  2          -0.167318                 0   
4                  0          -1.207850                 0   

   Work_Experience_Score  
0                   16.0  
1                   19.5  
2                   17.5  
3                   16.5  
4                   20.0  
```

In [36]:

```python
#ML libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

#Feauture Defining
X = df_cleaned.drop(columns=["Starting_Salary"])  # All features except the target
y = df_cleaned["Starting_Salary"]  # Target variable

#Split Testing 80/20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
    "Neural Network (MLP)": MLPRegressor(hidden_layer_sizes=(64, 32), max_iter=500, rand
}
```

```
#Evaluate the models
results = {}

for model_name, model in models.items():
    # Train model
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)

    # Evaluate model performance
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)

    # Store results
    results[model_name] = {"RMSE": rmse, "R² Score": r2}

#DataFrame
results_df = pd.DataFrame(results).T

# Display
print(results_df)
```

```
                          RMSE  R² Score
Linear Regression     1.010500 -0.007729
Random Forest         1.022857 -0.032525
Gradient Boosting     1.023274 -0.033368
Neural Network (MLP)  1.119171 -0.236129
```

Setting Up the Model for Importing and Use

In [39]:
```
import joblib

#Using R score for best results
best_model_name = results_df["R² Score"].idxmax()
best_model = models[best_model_name]

# Saving model
joblib.dump(best_model, "best_model.pkl")
print(f"Best model '{best_model_name}' saved successfully.")
```

```
Best model 'Linear Regression' saved successfully.
```

In [ ]: