

```
import pandas as pd

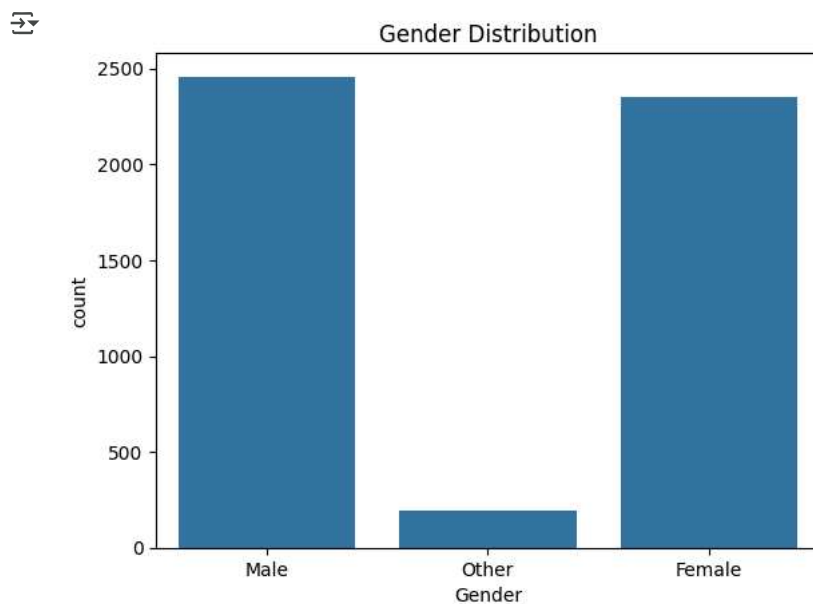
df = pd.read_csv('education_career_success.csv')
df.drop(columns=['Student_ID'], inplace=True)
df['Gender'] = df['Gender'].str.title().str.strip()
df['Field_of_Study'] = df['Field_of_Study'].str.title().str.strip()
df['Current_Job_Level'] = df['Current_Job_Level'].str.title().str.strip()
df['Entrepreneurship'] = df['Entrepreneurship'].str.title().str.strip()
df
```

	Age	Gender	High_School_GPA	SAT_Score	University_Ranking	University_GPA	Field_of_Study	Internships_Completed	Projects_Compl
0	24	Male	3.58	1052	291	3.96	Arts	3	
1	21	Other	2.52	1211	112	3.63	Law	4	
2	28	Female	3.42	1193	715	2.63	Medicine	4	
3	25	Male	2.43	1497	170	2.81	Computer Science	3	
4	22	Male	2.08	1012	599	2.48	Engineering	4	
...	...	...	...	...	...	...	...	...	...
4995	26	Female	2.44	1258	776	2.44	Arts	3	
4996	18	Female	3.94	1032	923	3.73	Law	0	
4997	19	Female	3.45	1299	720	2.52	Law	3	
4998	19	Male	2.70	1038	319	3.94	Law	1	
4999	23	Female	2.19	1145	82	3.19	Computer Science	2	

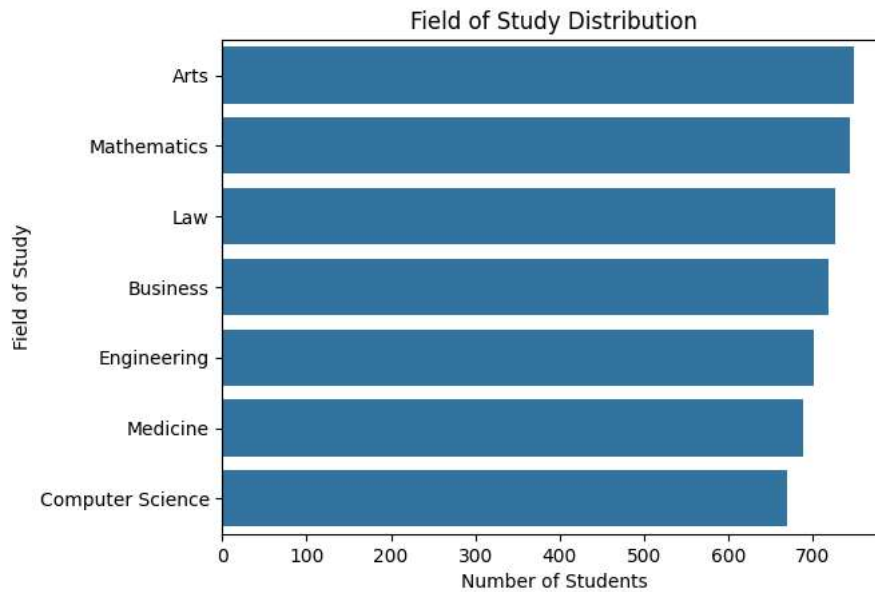
5000 rows × 19 columns

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

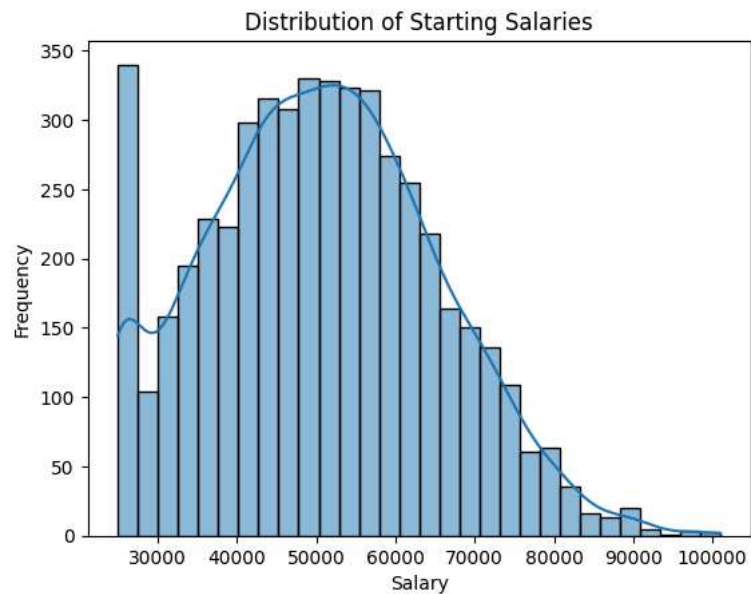
sns.countplot(x='Gender', data=df)
plt.title('Gender Distribution')
plt.show()
```




```
sns.countplot(y='Field_of_Study', data=df, order=df['Field_of_Study'].value_counts().index)
plt.title('Field of Study Distribution')
plt.xlabel('Number of Students')
plt.ylabel('Field of Study')
plt.show()
```



```
sns.histplot(df['Starting_Salary'], bins=30, kde=True)
plt.title('Distribution of Starting Salaries')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.show()
```

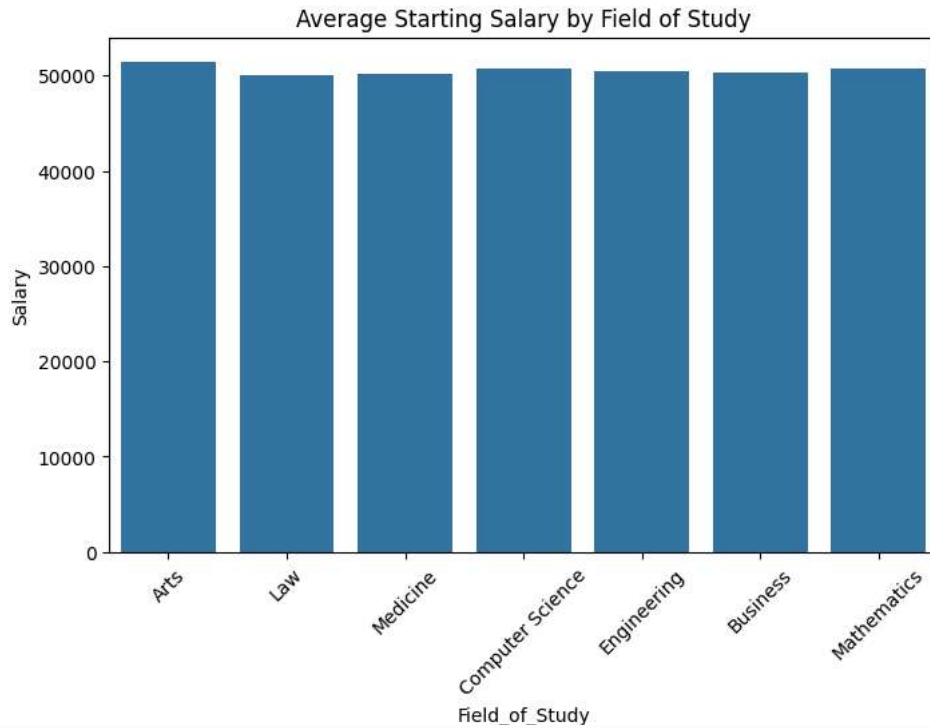


```
plt.figure(figsize=(8, 5))
sns.barplot(x='Field_of_Study', y='Starting_Salary', data=df, ci=None)
plt.xticks(rotation=45)
plt.title('Average Starting Salary by Field of Study')
plt.ylabel('Salary')
plt.show()
```

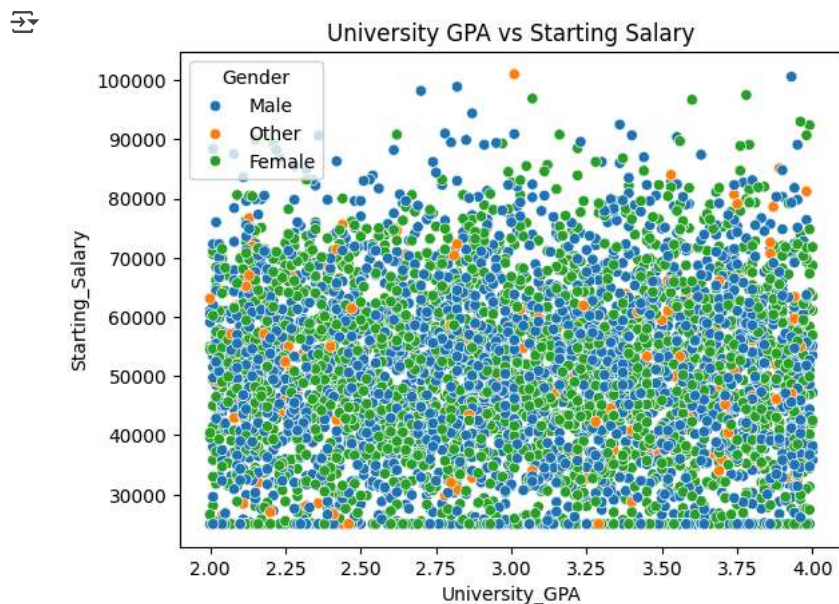
 /tmp/ipython-input-40-1575625863.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

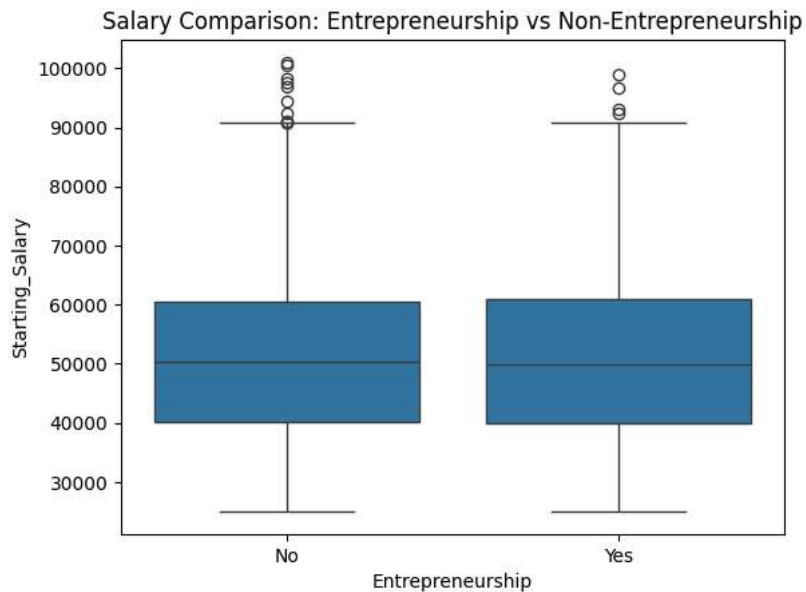
```
sns.barplot(x='Field_of_Study', y='Starting_Salary', data=df, ci=None)
```



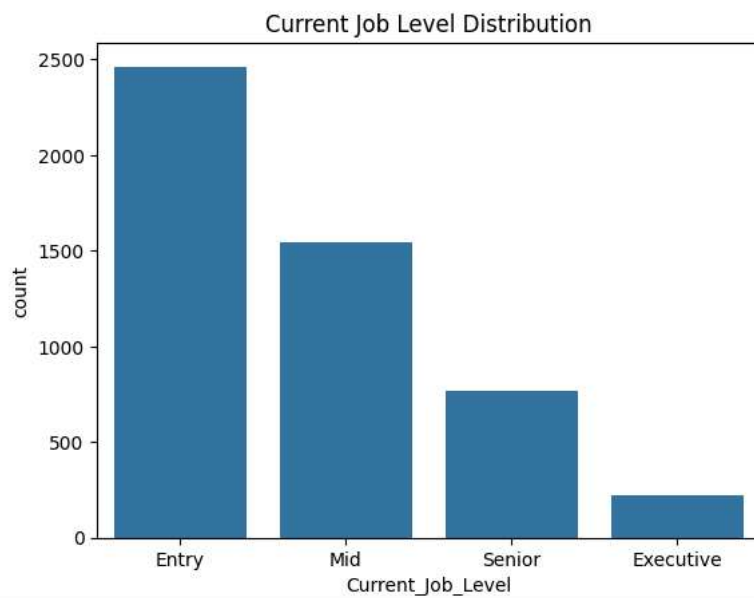
```
sns.scatterplot(x='University_GPA', y='Starting_Salary', hue='Gender', data=df)
plt.title('University GPA vs Starting Salary')
plt.show()
```



```
sns.boxplot(x='Entrepreneurship', y='Starting_Salary', data=df)
plt.title('Salary Comparison: Entrepreneurship vs Non-Entrepreneurship')
plt.show()
```



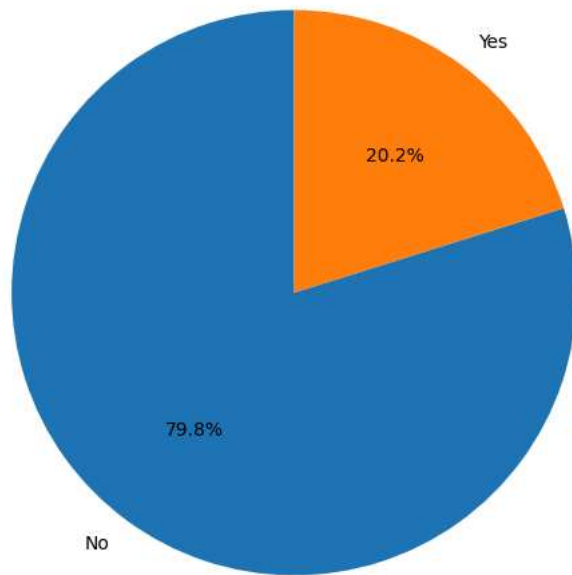
```
sns.countplot(x='Current_Job_Level', data=df, order=['Entry', 'Mid', 'Senior', 'Executive'])
plt.title('Current Job Level Distribution')
plt.show()
```



```
entrepreneurship_counts = df['Entrepreneurship'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(entrepreneurship_counts, labels=entrepreneurship_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Entrepreneurship Participation')
plt.axis('equal')
plt.show()
```



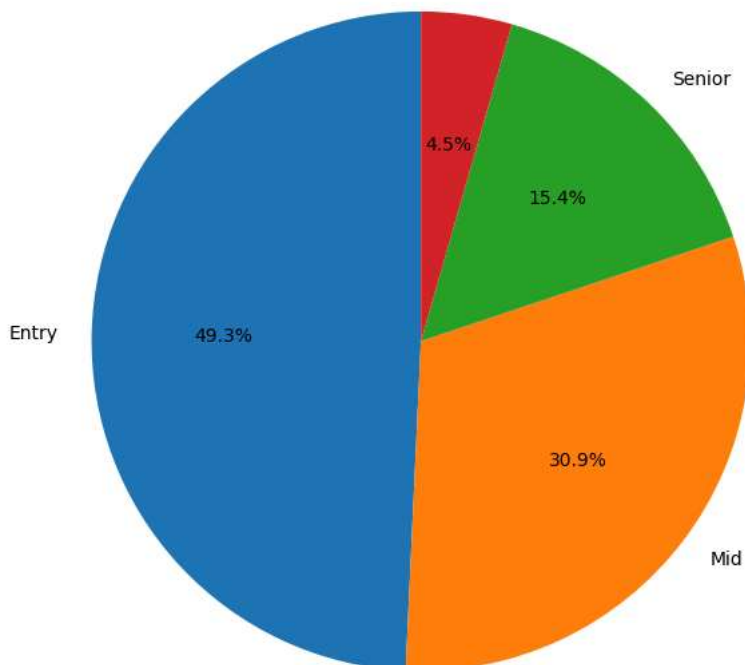
Entrepreneurship Participation



```
job_level_counts = df['Current_Job_Level'].value_counts()
plt.figure(figsize=(7, 7))
plt.pie(job_level_counts, labels=job_level_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Current Job Level Proportions')
plt.axis('equal')
plt.show()
```



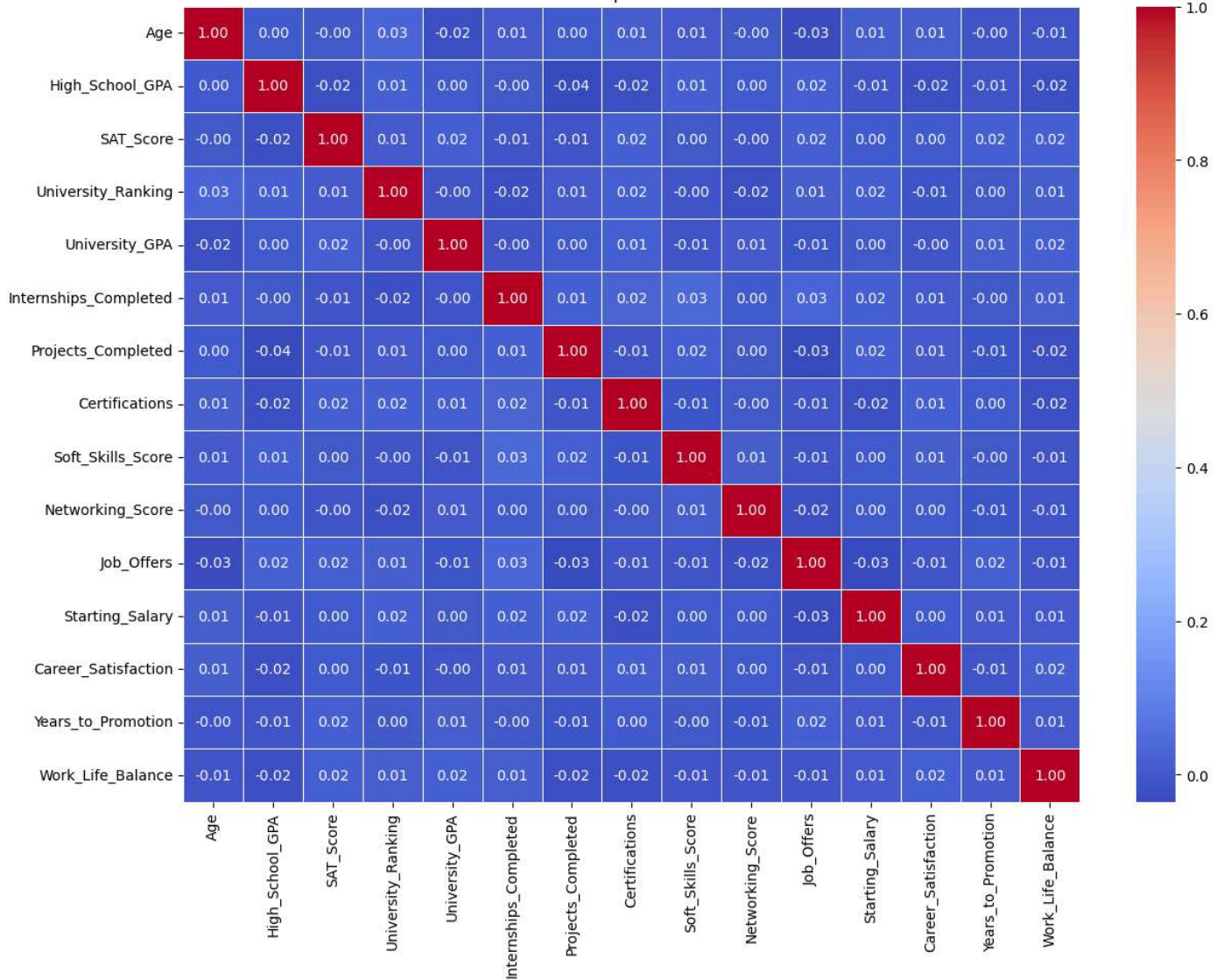
Current Job Level Proportions



```
plt.figure(figsize=(14, 10))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```



Correlation Heatmap of Numerical Features



```
from sklearn.preprocessing import LabelEncoder
```

```
df_model = df.copy()
df_model['Entrepreneurship'] = df_model['Entrepreneurship'].map({'Yes': 1, 'No': 0})
df_model['Gender'] = LabelEncoder().fit_transform(df_model['Gender'])
df_model['Field_of_Study'] = LabelEncoder().fit_transform(df_model['Field_of_Study'])
df_model['Current_Job_Level'] = LabelEncoder().fit_transform(df_model['Current_Job_Level'])
```

```
X = df_model.drop(columns=['Starting_Salary'])
y = df_model['Starting_Salary']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
```

```

model = RandomForestRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("Mean Absolute Error:", mae)
print("Root Mean Squared Error:", rmse)

```

↗ Mean Absolute Error: 12021.87  
Root Mean Squared Error: 14851.049714952813

```
df['Starting_Salary'].describe()
```

↗

	Starting_Salary
count	5000.000000
mean	50563.540000
std	14494.958207
min	25000.000000
25%	40200.000000
50%	50300.000000
75%	60500.000000
max	101000.000000

df.describe()

```

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

print("MAE (Linear):", mean_absolute_error(y_test, y_pred_lr))
print("RMSE (Linear):", np.sqrt(mean_squared_error(y_test, y_pred_lr)))

```

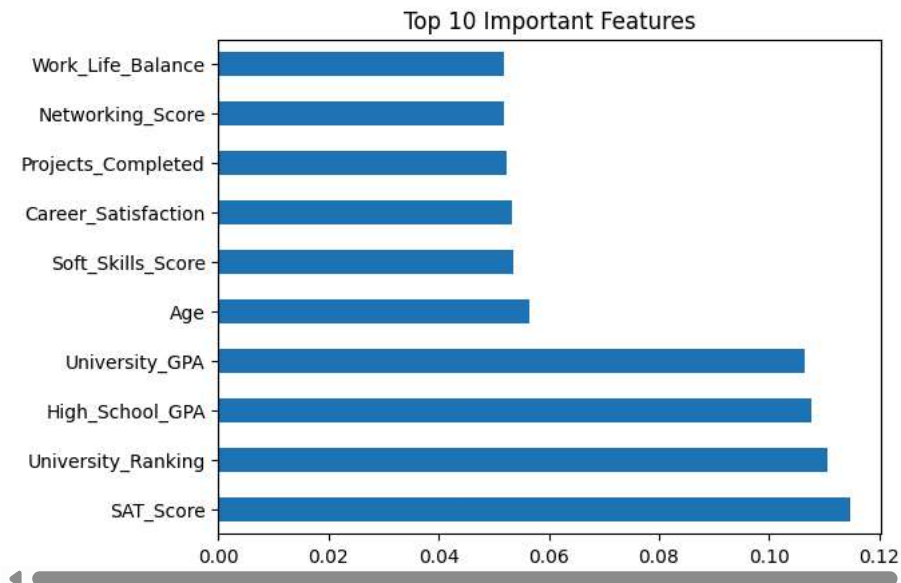
↗ MAE (Linear): 11872.66510563246  
RMSE (Linear): 14645.842188086348

```

import seaborn as sns
import matplotlib.pyplot as plt

pd.Series(model.feature_importances_, index=X.columns).nlargest(10).plot(kind='barh')
plt.title('Top 10 Important Features')
plt.show()

```



```
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

xgb = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=5, random_state=42)
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)
```

```
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))

print("MAE (XGBoost):", mae_xgb)
print("RMSE (XGBoost):", rmse_xgb)
```



```
MAE (XGBoost): 12195.11375
RMSE (XGBoost): 15065.451908191233
```

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

feature_importance = pd.Series(xgb.feature_importances_, index=X.columns)
feature_importance.nlargest(10).plot(kind='barh')
plt.title('Top 10 Features (XGBoost)')
plt.show()
```



```
import joblib
```

```
joblib.dump(model, 'salary_predictor.pkl')
```

```
['salary_predictor.pkl']
```

```
import matplotlib.pyplot as plt
```

```
models = ['Linear Regression', 'Random Forest', 'XGBoost']
```

```
mae_scores = [11873, 12068, 12195]
```

```
rmse_scores = [14646, 14836, 15065]
```

```
x = range(len(models))
```

```
bar_width = 0.35
```

```
plt.figure(figsize=(10, 6))
```

```
plt.bar(x, mae_scores, width=bar_width, label='MAE', color='skyblue')
```

```
plt.bar([p + bar_width for p in x], rmse_scores, width=bar_width, label='RMSE', color='salmon')
```

```
plt.xticks([p + bar_width/2 for p in x], models)
```

```
plt.ylabel('Error ( $\bar{x}$ )')
```

```
plt.title('Model Performance Comparison')
```

```
plt.legend()
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
plt.tight_layout()
```

```
plt.show()
```

