```
# ======================================
# 1. Data Loading
# ======================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Load dataset
df = pd.read_csv("/content/drive/MyDrive/DS160/Final Project/AI_Impact_on_Jobs_2030.csv")

df.head()
```

	Job_Title	Average_Salary	Years_Experience	Education_Level	AI_Exposure_Index	Tech_Growth_Factor	Au
0	Security Guard	45795	28	Master's	0.18	1.28	
1	Research Scientist	133355	20	PhD	0.62	1.11	
2	Construction Worker	146216	2	High School	0.86	1.18	
3	Software Engineer	136530	13	PhD	0.39	0.68	
4	Financial Analyst	70397	22	High School	0.52	1.46	

```
# Check dataset info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 18 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Job_Title                    3000 non-null   object
 1   Average_Salary               3000 non-null   int64
 2   Years_Experience             3000 non-null   int64
 3   Education_Level              3000 non-null   object
 4   AI_Exposure_Index            3000 non-null   float64
 5   Tech_Growth_Factor           3000 non-null   float64
 6   Automation_Probability_2030  3000 non-null   float64
 7   Risk_Category                3000 non-null   object
 8   Skill_1                      3000 non-null   float64
 9   Skill_2                      3000 non-null   float64
 10  Skill_3                      3000 non-null   float64
 11  Skill_4                      3000 non-null   float64
 12  Skill_5                      3000 non-null   float64
 13  Skill_6                      3000 non-null   float64
 14  Skill_7                      3000 non-null   float64
 15  Skill_8                      3000 non-null   float64
 16  Skill_9                      3000 non-null   float64
 17  Skill_10                     3000 non-null   float64
dtypes: float64(13), int64(2), object(3)
memory usage: 422.0+ KB
```

```
# Check shape
df.shape
```

```
(3000, 18)
```

```
df.isnull().sum()
```

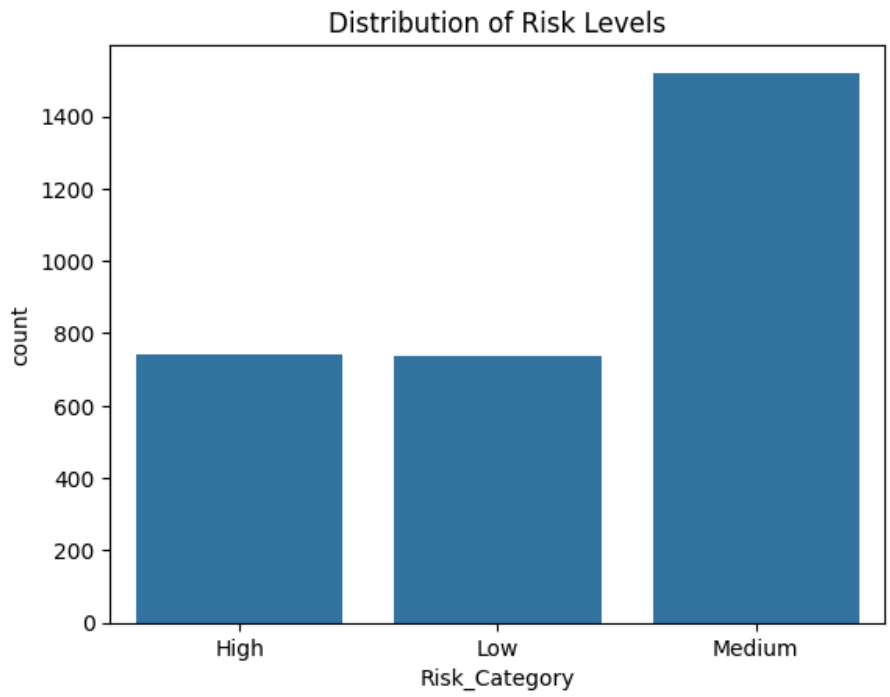
	0
Job_Title	0
Average_Salary	0
Years_Experience	0
Education_Level	0
AI_Exposure_Index	0
Tech_Growth_Factor	0
Automation_Probability_2030	0
Risk_Category	0
Skill_1	0
Skill_2	0
Skill_3	0
Skill_4	0
Skill_5	0
Skill_6	0
Skill_7	0
Skill_8	0
Skill_9	0
Skill_10	0

dtype: int64

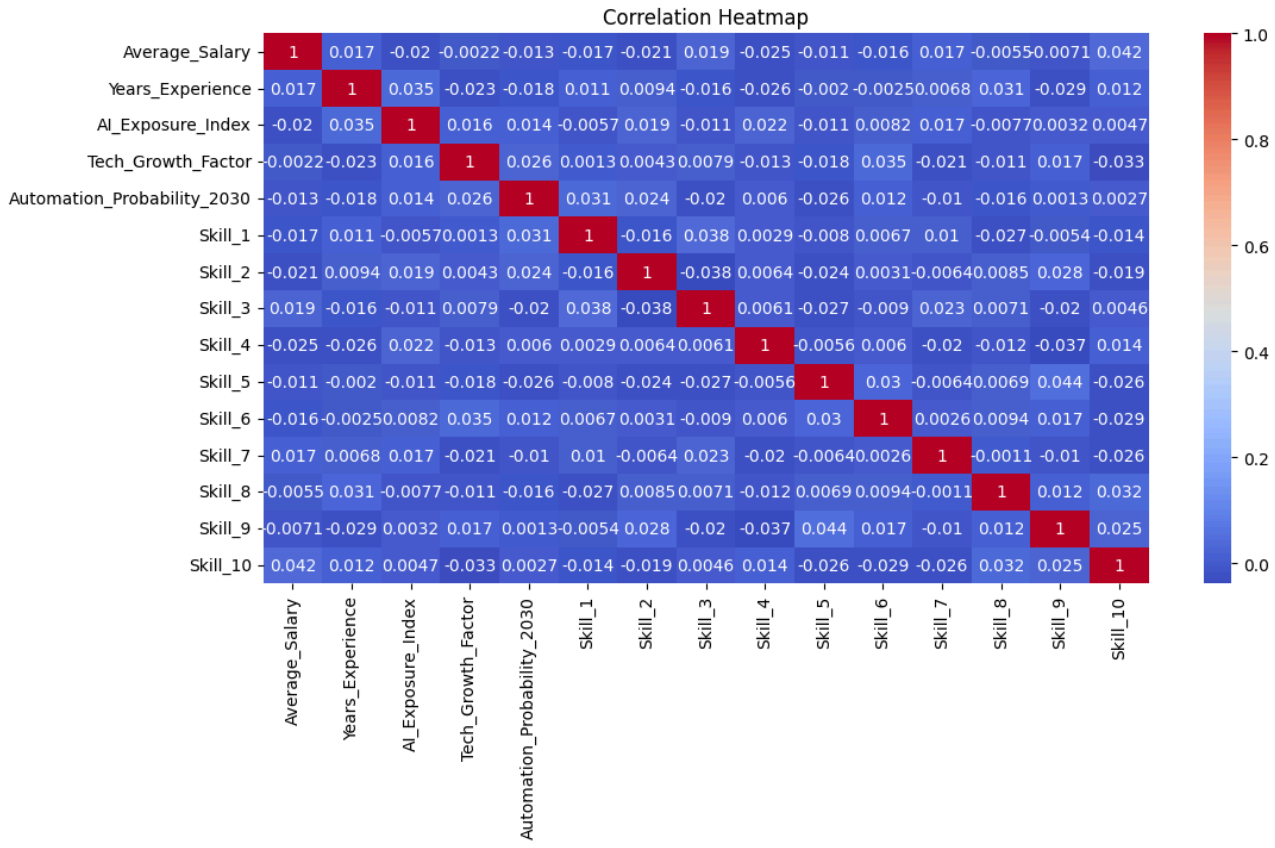
```
df.describe(include="all")
```

	Job_Title	Average_Salary	Years_Experience	Education_Level	AI_Exposure_Index	Tech_Growth_Factor
count	3000	3000.000000	3000.000000	3000	3000.000000	3000.000000
unique	20	NaN	NaN	4	NaN	NaN
top	Software Engineer	NaN	NaN	High School	NaN	NaN
freq	175	NaN	NaN	784	NaN	NaN
mean	NaN	89372.279000	14.677667	NaN	0.501283	0.995343
std	NaN	34608.088767	8.739788	NaN	0.284004	0.287669
min	NaN	30030.000000	0.000000	NaN	0.000000	0.500000
25%	NaN	58640.000000	7.000000	NaN	0.260000	0.740000
50%	NaN	89318.000000	15.000000	NaN	0.500000	1.000000
75%	NaN	119086.500000	22.000000	NaN	0.740000	1.240000
max	NaN	149798.000000	29.000000	NaN	1.000000	1.500000

```
sns.countplot(data=df, x="Risk_Category")
plt.title("Distribution of Risk Levels")
plt.show()
```

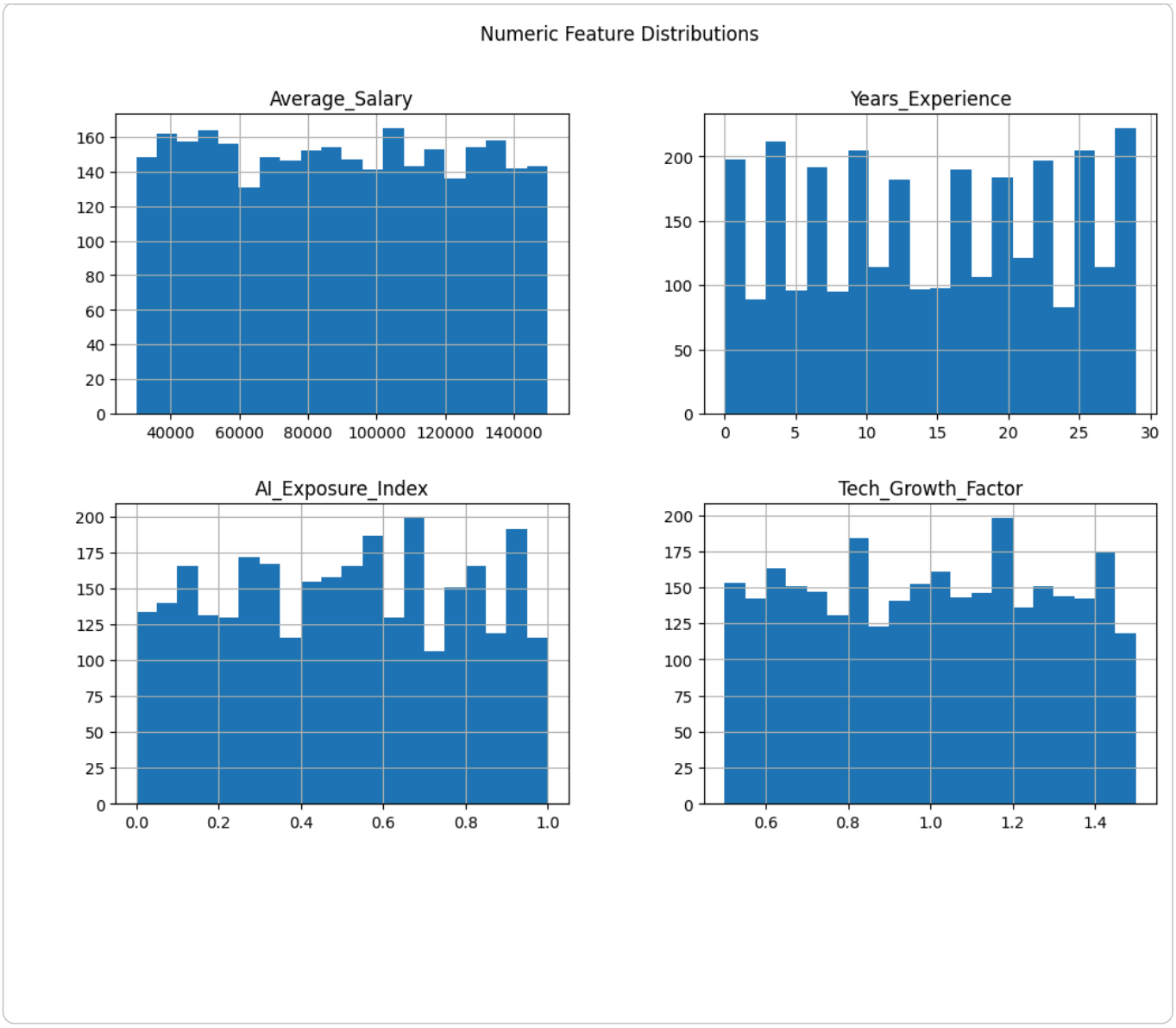



```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```




```
numeric_cols = df.select_dtypes(include=np.number).columns[:4]

df[numeric_cols].hist(figsize=(12,8), bins=20)
plt.suptitle("Numeric Feature Distributions")
plt.show()
```

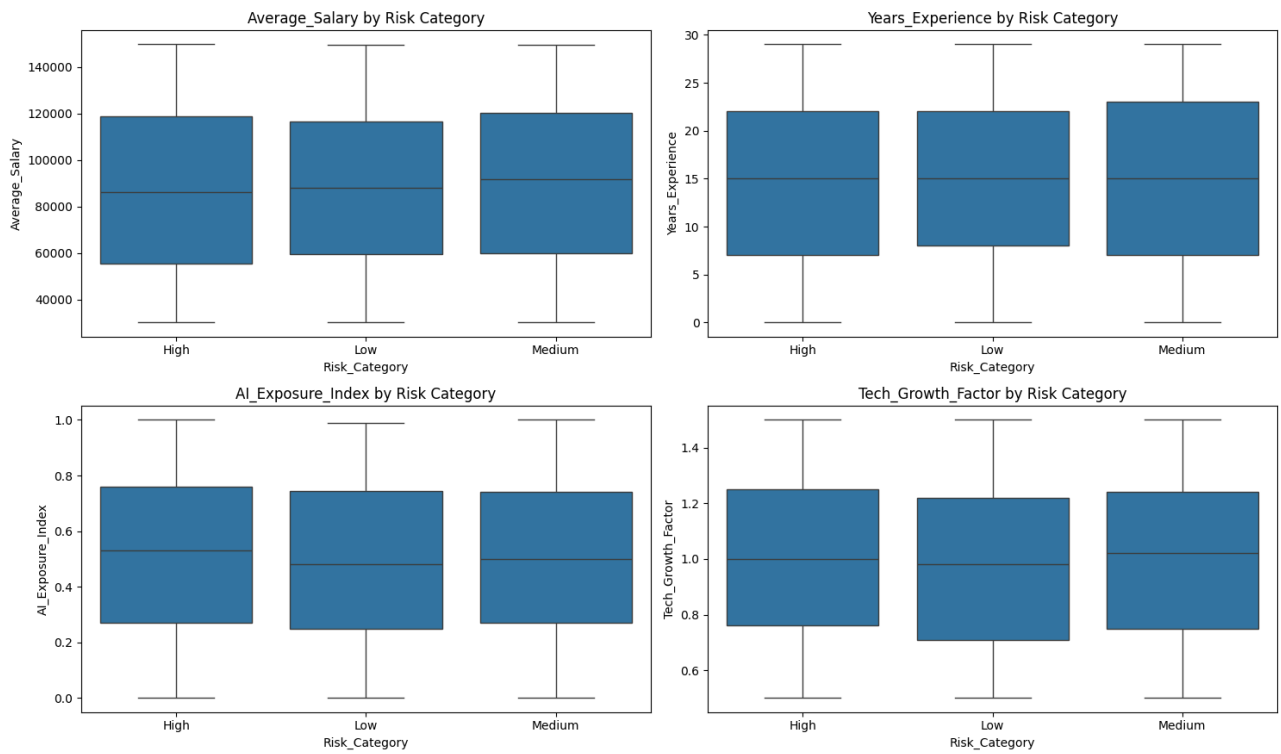



```
plt.figure(figsize=(15, 10))

numeric_cols_for_boxplot = ['Average_Salary', 'Years_Experience', 'AI_Exposure_Index', 'Tech_Growth_Factor

for i, col in enumerate(numeric_cols_for_boxplot):
    plt.subplot(2, 2, i + 1) # Create a 2x2 grid for 4 plots
    sns.boxplot(data=df, x="Risk_Category", y=col)
    plt.title(f'{col} by Risk Category')

plt.suptitle("Boxplots of Major Features by Risk Category", y=1.02) # Adjust suptitle position
plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout to prevent title overlap
plt.show()
```




```
df_encoded = pd.get_dummies(df, drop_first=True)
df_encoded.head()
```

	Average_Salary	Years_Experience	AI_Exposure_Index	Tech_Growth_Factor	Automation_Probability_2030	Sk
0	45795	28	0.18	1.28	0.85	
1	133355	20	0.62	1.11	0.05	
2	146216	2	0.86	1.18	0.81	
3	136530	13	0.39	0.68	0.60	
4	70397	22	0.52	1.46	0.64	

5 rows × 39 columns

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df["Risk_Category_Encoded"] = label_encoder.fit_transform(df["Risk_Category"])
df["Risk_Category_Encoded"].head()
```

	Risk_Category_Encoded
0	0
1	1
2	0
3	2
4	2

dtype: int64

```
y = df["Risk_Category_Encoded"]

X = df.drop(columns=["Risk_Category", "Risk_Category_Encoded"])
X = pd.get_dummies(X, drop_first=True)   # encode remaining categorical features

X.head()
```

	Average_Salary	Years_Experience	AI_Exposure_Index	Tech_Growth_Factor	Automation_Probability_2030	Sk
0	45795	28	0.18	1.28	0.85	
1	133355	20	0.62	1.11	0.05	
2	146216	2	0.86	1.18	0.81	
3	136530	13	0.39	0.68	0.60	
4	70397	22	0.52	1.46	0.64	

5 rows × 37 columns

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y)
```

```
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
log_model = LogisticRegression(max_iter=1000, multi_class="auto")
log_model.fit(X_train_scaled, y_train)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class
  warnings.warn(
```

LogisticRegression

```
LogisticRegression(max_iter=1000, multi_class='auto')
```

```
y_pred = log_model.predict(X_test_scaled)

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```