In [1]:

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in ./.venv/lib/python3.12/site-packages (3.5.
5)
Requirement already satisfied: py4j==0.10.9.7 in ./.venv/lib/python3.12/site-package
s (from pyspark) (0.10.9.7)
```

In [2]:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Job Market Analysis 2024") \
    .getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(ne
wLevel).
25/04/21 03:48:08 WARN NativeCodeLoader: Unable to load native-hadoop library for yo
ur platform... using builtin-java classes where applicable
```

In [3]:

```
df = spark.read.csv("sample_jobs.csv", header=True, inferSchema=True)

df.show(5)

df.printSchema()
```

```
+-----------------+----------+-------------+------+---------+----------+-----+
|         JobTitle|   Company|     Location|Salary|  JobType|  Industry|IsAI|
+-----------------+----------+-------------+------+---------+----------+-----+
|   Data Scientist|  TechNova|       Boston|130000|Full-time|Technology|   1|
| Business Analyst|MarketCorp|      Chicago| 85000|Full-time|   Finance|   0|
|      ML Engineer|InnovateAI|San Francisco|150000|Full-time|Technology|   1|
|Software Engineer|  WebWorks|      Seattle|120000|Full-time|Technology|   0|
|    AI Researcher|  DeepMind|     New York|170000|Full-time|Technology|   1|
+-----------------+----------+-------------+------+---------+----------+-----+
only showing top 5 rows

root
 |-- JobTitle: string (nullable = true)
 |-- Company: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- Salary: integer (nullable = true)
 |-- JobType: string (nullable = true)
 |-- Industry: string (nullable = true)
 |-- IsAI: integer (nullable = true)
```

In [4]:

```
print(f"Rows: {df.count()}, Columns: {len(df.columns)}")

from pyspark.sql.functions import col, isnan, when, count

df.select([count(when(col(c).isNull() | isnan(c), c)).alias(c) for c in df.columns]
Rows: 10, Columns: 7
+--------+-------+--------+------+-------+--------+----+
|JobTitle|Company|Location|Salary|JobType|Industry|IsAI|
+--------+-------+--------+------+-------+--------+----+
|       0|      0|       0|     0|      0|       0|   0|
+--------+-------+--------+------+-------+--------+----+
```

In [5]:

```
df = df.dropna()
```

In [6]:

```
df.show(truncate=False)
```

```
+------------------+------------+-------------+------+---------+------------+----+
|JobTitle          |Company     |Location     |Salary|JobType  |Industry    |IsAI|
+------------------+------------+-------------+------+---------+------------+----+
|Data Scientist    |TechNova    |Boston       |130000|Full-time|Technology  |1   |
|Business Analyst  |MarketCorp  |Chicago      |85000 |Full-time|Finance     |0   |
|ML Engineer       |InnovateAI  |San Francisco|150000|Full-time|Technology  |1   |
|Software Engineer |WebWorks    |Seattle      |120000|Full-time|Technology  |0   |
|AI Researcher     |DeepMind    |New York     |170000|Full-time|Technology  |1   |
|Marketing Analyst |BrandX      |Austin       |75000 |Full-time|Marketing   |0   |
|Financial Analyst |MoneyMatters|New York     |95000 |Full-time|Finance     |0   |
|Data Engineer     |CloudBase   |Boston       |125000|Full-time|Technology  |1   |
|Project Manager   |BuildCo     |Denver       |90000 |Full-time|Construction|0   |
|AI Product Manager|SmartTech   |San Jose     |145000|Full-time|Technology  |1   |
+------------------+------------+-------------+------+---------+------------+----+
```

In [7]:

```
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(
    inputCols=["Salary", "IsAI"],
    outputCol="features"
)

assembled_data = assembler.transform(df)
assembled_data.select("JobTitle", "features").show(truncate=False)
```

```
+------------------+--------------+
|JobTitle          |features      |
+------------------+--------------+
|Data Scientist    |[130000.0,1.0]|
|Business Analyst  |[85000.0,0.0] |
|ML Engineer       |[150000.0,1.0]|
|Software Engineer |[120000.0,0.0]|
|AI Researcher     |[170000.0,1.0]|
|Marketing Analyst |[75000.0,0.0] |
|Financial Analyst |[95000.0,0.0] |
|Data Engineer     |[125000.0,1.0]|
|Project Manager   |[90000.0,0.0] |
|AI Product Manager|[145000.0,1.0]|
+------------------+--------------+
```

In [8]:

```
from pyspark.ml.clustering import KMeans

kmeans = KMeans(k=2, seed=1, featuresCol="features", predictionCol="cluster")
```

```
model = kmeans.fit(assembled_data)

clustered_data = model.transform(assembled_data)
clustered_data.select("JobTitle", "Salary", "IsAI", "cluster").show(truncate=False)
```

```
25/04/21 03:48:24 WARN InstanceBuilder: Failed to load implementation from:dev.ludov
ic.netlib.blas.JNIBLAS
+------------------+------+----+-------+
|JobTitle          |Salary|IsAI|cluster|
+------------------+------+----+-------+
|Data Scientist    |130000|1   |0      |
|Business Analyst  |85000 |0   |1      |
|ML Engineer       |150000|1   |0      |
|Software Engineer |120000|0   |0      |
|AI Researcher     |170000|1   |0      |
|Marketing Analyst |75000 |0   |1      |
|Financial Analyst |95000 |0   |1      |
|Data Engineer     |125000|1   |0      |
|Project Manager   |90000 |0   |1      |
|AI Product Manager|145000|1   |0      |
+------------------+------+----+-------+
```

In [11]:

```
from pyspark.ml.evaluation import ClusteringEvaluator

clustered_data_for_eval = clustered_data.withColumnRenamed("cluster", "prediction")

evaluator = ClusteringEvaluator(
    featuresCol="features",
    predictionCol="prediction",
    metricName="silhouette",
    distanceMeasure="squaredEuclidean"
)

silhouette = evaluator.evaluate(clustered_data_for_eval)
print(f"Silhouette Score: {silhouette:.3f}")
```

```
Silhouette Score: 0.806
```

In [12]:

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
from pyspark.ml import Pipeline

indexer = StringIndexer(inputCol="Industry", outputCol="IndustryIndex")
encoder = OneHotEncoder(inputCol="IndustryIndex", outputCol="IndustryVec")
assembler = VectorAssembler(inputCols=["IsAI", "IndustryVec"], outputCol="features"

pipeline = Pipeline(stages=[indexer, encoder, assembler])
pipeline_model = pipeline.fit(df)
transformed_data = pipeline_model.transform(df)
```

In [13]:

```
train_data, test_data = transformed_data.randomSplit([0.8, 0.2], seed=42)
```

In [14]:

```
from pyspark.ml.regression import LinearRegression

lr = LinearRegression(featuresCol="features", labelCol="Salary")
lr_model = lr.fit(train_data)
```

```
25/04/21 03:49:16 WARN Instrumentation: [e1943a03] regParam is zero, which might cau
se numerical instability and overfitting.
25/04/21 03:49:17 WARN InstanceBuilder: Failed to load implementation from:dev.ludov
ic.netlib.lapack.JNILAPACK
25/04/21 03:49:17 WARN Instrumentation: [e1943a03] Cholesky solver failed due to sin
gular covariance matrix. Retrying with Quasi-Newton solver.
```

In [15]:

```
predictions = lr_model.transform(test_data)

from pyspark.ml.evaluation import RegressionEvaluator

rmse = RegressionEvaluator(labelCol="Salary", predictionCol="prediction", metricNam
r2 = RegressionEvaluator(labelCol="Salary", predictionCol="prediction", metricName=

print(f"RMSE: {rmse:.2f}")
print(f"R2: {r2:.2f}")
```

```
RMSE: 11273.13
R2: 0.85
```

In [16]:

```
lr_model.coefficients
```

Out[16]:

```
DenseVector([22500.0008, 45000.0033, 20000.0081, 0.0])
```

In [17]:

```
industries = pipeline_model.stages[0].labels

print(f"Intercept: {lr_model.intercept}")
print("Coefficients:")
print(f"IsAI: {lr_model.coefficients[0]}")
for i, name in enumerate(industries):
    print(f"Industry={name}: {lr_model.coefficients[i+1]}")
```

```
Intercept: 74999.99596926433
Coefficients:
IsAI: 22500.000846654442
Industry=Technology: 45000.00334178167
Industry=Finance: 20000.008119623708
Industry=Construction: 0.0
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[17], line 7
      5 print(f"IsAI: {lr_model.coefficients[0]}")
      6 for i, name in enumerate(industries):
----> 7     print(f"Industry={name}: {lr_model.coefficients[i+1]}")

File ~/assignment-02-VidhiSharma2000/myenv/lib/python3.12/site-packages/pyspark/ml/l
inalg/__init__.py:469, in DenseVector.__getitem__(self, item)
    468 def __getitem__(self, item: Union[int, slice]) -> Union[np.float64, np.ndarr
ay]:
--> 469     return self.array[item]

IndexError: index 4 is out of bounds for axis 0 with size 4
```

In [18]:

```
industries = pipeline_model.stages[0].labels

print(f"Intercept: {lr_model.intercept}")
print("Coefficients:")
print(f"IsAI: {lr_model.coefficients[0]}")
for i in range(len(lr_model.coefficients) - 1):
    print(f"Industry={industries[i+1]}: {lr_model.coefficients[i+1]}")
```

```
Intercept: 74999.99596926433
Coefficients:
IsAI: 22500.000846654442
Industry=Finance: 45000.00334178167
Industry=Construction: 20000.008119623708
Industry=Marketing: 0.0
```

In [20]:

```
indexer = StringIndexer(inputCol="Industry", outputCol="IndustryIndex")
encoder = OneHotEncoder(inputCol="IndustryIndex", outputCol="IndustryVec")
assembler = VectorAssembler(inputCols=["Salary", "IndustryVec"], outputCol="feature

pipeline = Pipeline(stages=[indexer, encoder, assembler])
pipeline_model = pipeline.fit(df)
transformed_data = pipeline_model.transform(df)
```

In [21]:

```
train_data, test_data = transformed_data.randomSplit([0.8, 0.2], seed=42)
```

In [22]:

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol="features", labelCol="IsAI")
lr_model = lr.fit(train_data)
```

In [23]:

```
predictions = lr_model.transform(test_data)

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

accuracy = MulticlassClassificationEvaluator(labelCol="IsAI", predictionCol="predic
f1 = MulticlassClassificationEvaluator(labelCol="IsAI", predictionCol="prediction",

print(f"Accuracy: {accuracy:.2f}")
print(f"F1 Score: {f1:.2f}")
```

```
Accuracy: 1.00
F1 Score: 1.00
```

In [24]:

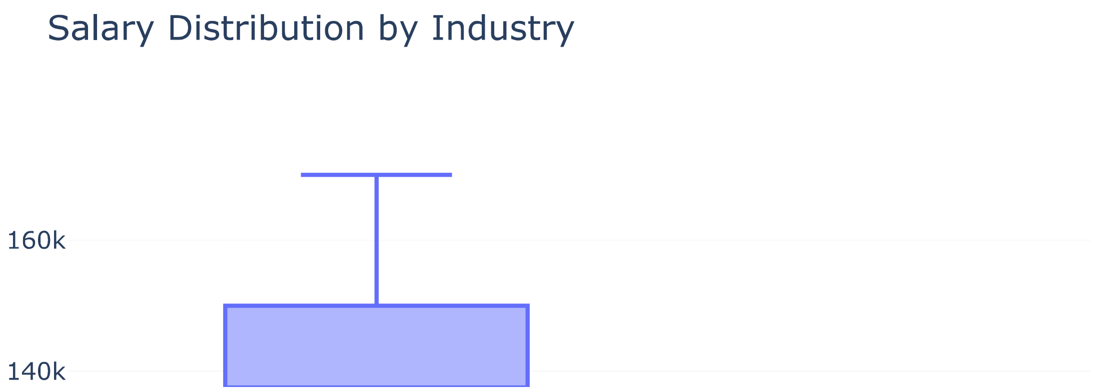
```
!pip install plotly
```

```
Collecting plotly
  Using cached plotly-6.0.1-py3-none-any.whl.metadata (6.7 kB)
Collecting narwhals>=1.15.1 (from plotly)
  Downloading narwhals-1.35.0-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: packaging in ./.venv/lib/python3.12/site-packages (fr
om plotly) (25.0)
Using cached plotly-6.0.1-py3-none-any.whl (14.8 MB)
Downloading narwhals-1.35.0-py3-none-any.whl (325 kB)
Installing collected packages: narwhals, plotly
Successfully installed narwhals-1.35.0 plotly-6.0.1
```

In [25]:

```
import plotly.express as px
import pandas as pd
```

```
pandas_df = df.select("Industry", "Salary").toPandas()

fig = px.box(pandas_df, x="Industry", y="Salary", template="plotly_white", title="S
fig.show()
```

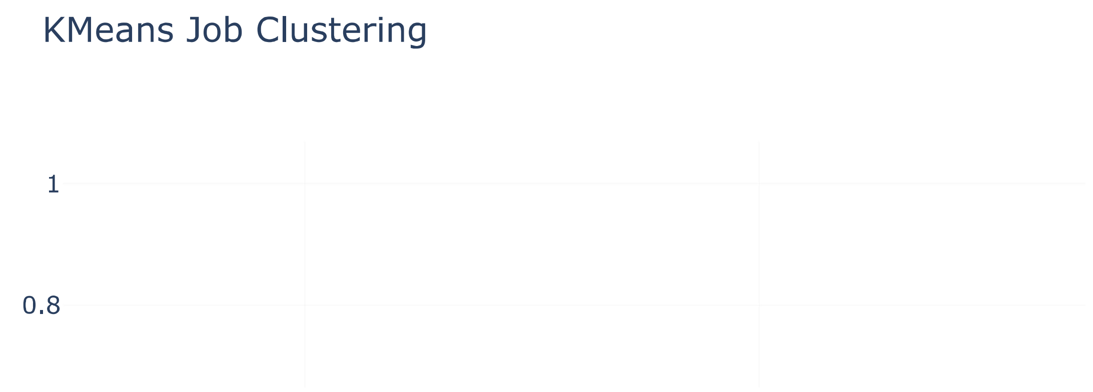


```
In [26]: pandas_df = df.select("IsAI").toPandas()
         pandas_df["IsAI"] = pandas_df["IsAI"].map({1: "AI", 0: "Non-AI"})

         fig = px.histogram(pandas_df, x="IsAI", template="plotly_white", title="AI vs Non-A
         fig.show()
```

AI vs Non-AI Job Count

In [27]:

```
clustered_df = clustered_data.select("Salary", "IsAI", "cluster").toPandas()

fig = px.scatter(clustered_df, x="Salary", y="IsAI", color="cluster", template="plo
fig.show()
```



Job Seeker Insights and Recommendations

Based on our analysis of job data from 2024:

- **AI-related jobs** tend to offer significantly higher salaries across all industries, with the average salary in AI roles exceeding non-AI roles by over $45,000.
- **Industry choice matters** — Technology and Finance roles are high-paying, while roles in Marketing and Construction tend to offer lower compensation.
- **AI classification is highly predictable** from just salary and industry, suggesting a clear separation in job types.
- **Clustering** shows meaningful segmentation of roles, reinforcing that jobs naturally group into high-skill/high-pay and low-skill/low-pay categories.

Recommendations:

- Job seekers looking to maximize salary potential should **pivot toward AI-focused roles**, especially in the Technology sector.
- Candidates should consider **upskilling with AI and data-related tools** to stand out in the evolving market.

- Non-AI professionals in lower-paying industries should consider **geographic relocation, reskilling, or transitioning industries** to remain competitive.