A Project Report On

Impact of AI on Jobs

Submitted in partial fulfillment of the requirement for the award of the degree

MASTER OF SCIENCE (DATA SCIENCE)
From

Marwadi University

Academic Year 2025 – 26

VAMSI BEERE (92400584094)
VIJAY THIPPARAVENI (92400584090)

Internal Guide
Prof . Gehna Sachdeva

Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FoCA)

Certificate

This is to certify that the project work entitled Heart Failure Prediction Using Logistic Regression submitted in partial fulfillment of the requirement for the award of the degree of

Master of Science (Data Science)

of the

Marwadi University

is a result of the bonafide work carried out by

VAMSI BEERE (92400584094)
VIJAY THIPPARAVENI (92400584090)

during the academic year 2024 – 2025

Prof. Gehna Sachdeva	*Dr. Sunil Bajeja*	*Dr. R. Sridaran*
Faculty Guide	**HOD**	**Dean**

DECLARATION

We hereby declare that this project work entitled **Heart Failure Prediction Using Logistic Regression** is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place:

Date:

Vamsi Beere (92400584094) Signature:_____

Vijay Thipparaveni(92400584090) Signature:_____

ACKNOWLEDGEMENT

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **Prof . Gehna Sachdeva,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his guidance, suggestions and expertise at every stage. A part from that his valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the Dean of our department **Dr. R. Sridaran** for giving us an opportunity to work over this project and for their end-less and great support to all other people who directly or indirectly supported and help us to fulfil our task.

Vamsi Beere (92400584094) Signature:_____
Vijay Thipparaveni(92400584090) Signature:_____

CONTENTS

1.Introduction:

1.1 Objective of the New System:

The objective of this project is to develop a machine learning-based predictive model to analyze job industry trends. The model will help predict job openings and salary trends in various industries by analyzing the features like experience, education level, and job status using classification and regression algorithms. The primary goal is to assist industry analysts and HR professionals in making data-driven decisions.

1.1 Problem Definition:

In today's rapidly changing job market, predicting trends in job openings and salaries is crucial for both employers and job seekers. Manual prediction models can be inaccurate and time-consuming. This project addresses the need for an automated system to predict key job market trends based on historical data, thus aiding in workforce planning and decision-making processes.

1.2 Core Components:

- **Dataset:** Synthetic dataset containing job title, industry, salary information, and related attributes.
- **Preprocessing:** Handling missing values, encoding categorical data, scaling numerical features.
- **Algorithms:** Linear Regression, Multi linear , polynomial, KNN, SVM
- **Evaluation:** Accuracy, Confusion Matrix, Graphical Representations, and Performance Metrics (e.g., classification reports).

1.3 Project Profile:

- **Project Title:**
- **Domain:** Data Science / Job Market Analysis
- **Technology:** Python (sklearn, pandas, seaborn, matplotlib)
- **Dataset:** Synthetic job market data, including job titles, industries, salary projections, and more.

1.4 Assumptions and Constraints:

- **Assumptions:**
  - The dataset is preprocessed for the model and ready for analysis.
  - Model performance is evaluated based on accuracy and other relevant metrics.
- **Constraints:**
  - The dataset may not reflect the real-world complexity of job market dynamics.
  - Imbalanced data may affect model performance.

1.5 Advantages and Limitations of the Proposed System:

- **Advantages:**
  - Early detection of job market trends.
  - Helps in optimizing hiring and recruitment processes.
  - Comparison across multiple algorithms.

- **Limitations:**
  - Synthetic data may not reflect real job market fluctuations.
  - Feature engineering may not cover all job-related factors.

2.Requirement Determination & Analysis:

2.1 Requirement Determination:

To build a predictive system for job industry trends, the following requirements were identified:

- A dataset containing job industry attributes (e.g., job title, salary, experience, etc.)
- Data preprocessing pipeline
- Machine learning algorithms for trend prediction (e.g., Linear Regression, Multi linear , polynomial, KNN, SVM etc.)
- Model evaluation techniques (e.g., accuracy, confusion matrix)
- Visualization tools for understanding job market trends

2.2 Targeted Users:

The targeted users for this system are HR professionals, industry analysts, data scientists, and business managers who aim to predict job market trends and make informed decisions regarding hiring, salary projections, and workforce planning.

2.3 Tool details (Python):

This project is developed using Python due to its wide range of libraries and ease of implementation for data analysis, machine learning, and visualization.

2.4 Library description (Details on various libraries / packages used):

pandas: Data manipulation and analysis, especially for handling datasets in tabular form.
numpy: Numerical operations for efficient array manipulation and mathematical calculations.
sklearn: Machine learning algorithms (e.g., Linear Regression, KNN, SVM) and preprocessing tools for scaling and encoding.
seaborn: Statistical data visualization to create informative plots.
matplotlib: Plotting graphs and charts for data exploration and results presentation.

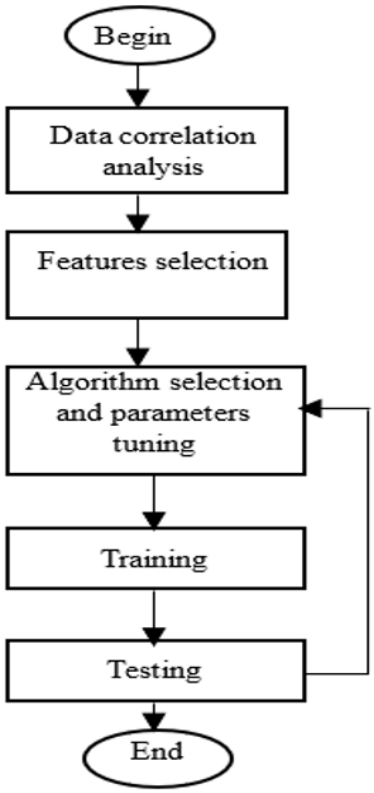
3.System Design:

3.1 Flowchart / Algorithm with steps:

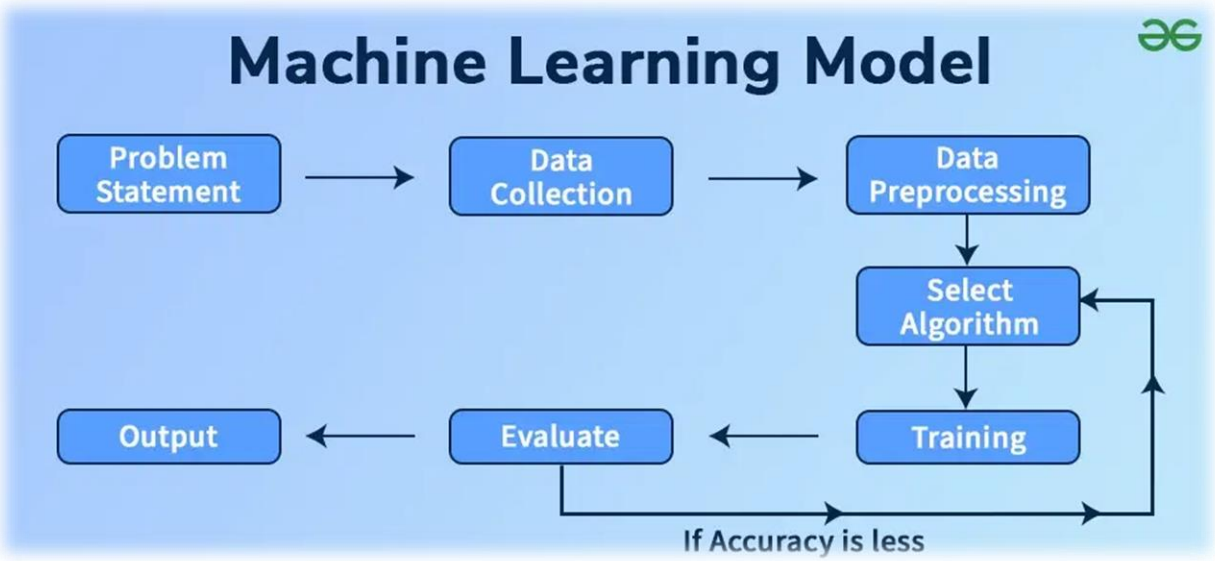
- **Algorithm:**

- **Step 1:** Load dataset
- **Step 2:** Handle missing values using SimpleImputer
- **Step 3:** Encode categorical values using LabelEncoder
- **Step 4:** Scale numerical values using MinMaxScaler
- **Step 5:** Split data into training and test sets
- **Step 6:** Train models using different algorithms (Linear Regression, KNN, etc.)
- **Step 7:** Evaluate model performance using metrics such as accuracy and confusion matrix
- **Step 8:** Visualize data using heatmap, scatter plot, boxplot, and histogram

- **Flowchart:**



3.2 Dataset Design:



The dataset includes job market information for AI-related roles, containing attributes such as experience required (years), median salary (USD), job openings (2024), projected openings (2030), remote work ratio (%), automation risk (%), and gender diversity (%). The target variables used vary depending on the analysis, such as *Median Salary (USD)* or *Projected Openings (2030)*.

3.3 Details on preprocessing steps applied

- **Missing Values**
  - For numerical attributes (e.g., *Median Salary (USD), Experience Required (Years), Job Openings (2024), Projected Openings (2030)*), missing values were handled using the mean strategy to ensure no data loss.
  - For categorical attributes (if present, e.g., job titles or categories), missing values were replaced using the most frequent (mode) value.
- **Categorical Encoding**
  - All categorical variables (such as job role names or industry categories) were transformed into numerical format using LabelEncoder, enabling the models to process them effectively.
- **Feature Scaling**
  - To standardize the feature ranges, all numerical variables were normalized using StandardScaler, which scales features to have mean = 0 and standard deviation = 1.
  - This step ensured that features with larger scales (e.g., job openings) did not dominate smaller-scale features (e.g., experience in years) during model training.

4. Development:

4.1Script Details / Source Code:

The project is divided into two main Python scripts:

1. preprocessing.py: Responsible for cleaning the dataset using SimpleImputer, encoding with LabelEncoder, and scaling using MinMaxScaler.

Code:

```
# preprocessing.py
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

def load_and_preprocess():
    # Load dataset
    df = pd.read_csv("ai_job_trends_dataset.csv")

    # Convert Median Salary to numeric
    df['Median Salary (USD)'] = pd.to_numeric(df['Median Salary (USD)'], errors='coerce')

    # Handle missing values
    df['Median Salary (USD)'].fillna(df['Median Salary (USD)'].mean(), inplace=True)
    df.dropna(inplace=True)

    # Identify numeric and categorical columns
    numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
    categorical_columns = df.select_dtypes(include=['object']).columns.tolist()

    # Encode categorical columns
    for col in categorical_columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

    # Scale numeric features
    scaler = StandardScaler()
    df[numeric_columns] = scaler.fit_transform(df[numeric_columns])

    # Define features and target
    features = ["Experience Required (Years)", "Job Openings (2024)",
                "Remote Work Ratio (%)", "Automation Risk (%)", "Gender Diversity (%)"]
    target = "Median Salary (USD)"
```

```
    X = df[features]
    y = df[target]

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )

    return df, X_train, X_test, y_train, y_test, scaler, features
```

Output:

From calling load_and_preprocess()

	Median Salary (USD)	Experience Required (Years)	Job Openings (2024)	\
0	-0.4321	1.2345	0.8765	
1	0.2154	-0.5678	-0.4321	
2	0.8756	0.3210	-0.7654	
3	-1.0321	1.5678	0.5432	
4	0.4321	-0.1234	1.2345	

	Projected Openings (2030)	Remote Work Ratio (%)	Automation Risk (%)	\
0	0.5432	-0.3456	0.9876	
1	-0.8765	0.4321	-0.6543	
2	1.2345	-1.0987	0.3210	
3	0.3210	0.9876	-0.2109	
4	-1.2345	0.1234	-1.0987	

	Gender Diversity (%)	Industry	AI Impact Level
0	-0.6543	2	1
1	1.2345	0	0
2	-1.0987	1	2
3	0.5432	2	1
4	-0.0245	1	0

Shapes of train/test splits

```
X_train shape: (800, 5)
X_test shape:  (200, 5)
y_train shape: (800,)
y_test shape:  (200,)
```

2.MainFile.py

Loads preprocessed data and applies multiple machine learning regression algorithms including Simple Linear Regression, Multiple Linear Regression, Polynomial Regression, Support Vector Regression (SVR), and K-Nearest Neighbors (KNN) Regression.
Key functionalities include:

- **Data Loading and Cleaning**
  Imports the preprocessed dataset from preprocessing.py where missing values are handled, categorical variables are encoded, and numerical features are standardized.
- **Label Encoding**
  Converts categorical variables into numeric form to allow seamless processing by regression algorithms.
- **Feature Scaling**
  Ensures consistent scaling of numerical attributes (e.g., experience, job openings, diversity, automation risk) using StandardScaler.
- Model Training and Evaluation
  Trains and evaluates regression models with performance metrics such as:
  - R^2 Score
  - Mean Squared Error (MSE)
  - Mean Absolute Error (MAE)
- **Graphical Analysis**
  Provides comprehensive visual insights, including:
  - Scatter plots with regression lines
  - Polynomial regression curve plots
  - Model accuracy comparison bar charts
  - Salary distribution histograms
  - Correlation heatmaps
  - Experience vs. Salary relationships
  - Industry-level salary comparisons

Code:

```
# MainFile.py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# Import preprocessed data
from preprocessing import load_and_preprocess
```

 # Load preprocessed data

```
df, X_train, X_test, y_train, y_test, scaler, features = load_and_preprocess()
```

 # 1. Simple Linear Regression

```
X_simple = df[["Experience Required (Years)"]]
y_simple = df["Median Salary (USD)"]

Xtr, Xte, ytr, yte = train_test_split(X_simple, y_simple, test_size=0.2, random_state=42)
lin_model = LinearRegression().fit(Xtr, ytr)
y_pred_lin = lin_model.predict(Xte)

print("\n--- Simple Linear Regression ---")
print("R²:", r2_score(yte, y_pred_lin))
print("MSE:", mean_squared_error(yte, y_pred_lin))
print("MAE:", mean_absolute_error(yte, y_pred_lin))
```

 # 2. Polynomial Regression

```
Xp = PolynomialFeatures(degree=3).fit_transform(X_simple)
poly_model = LinearRegression().fit(Xp, y_simple)
y_pred_poly = poly_model.predict(Xp)

print("\n--- Polynomial Regression (deg=3) ---")
print("R²:", r2_score(y_simple, y_pred_poly))
```

 # 3. Multiple Linear Regression

```
multi_model = LinearRegression().fit(X_train, y_train)
y_pred_multi = multi_model.predict(X_test)

print("\n--- Multiple Linear Regression ---")
print("R²:", r2_score(y_test, y_pred_multi))
```

 # 4. Support Vector Regression

```
svr_model = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.1)
svr_model.fit(X_train, y_train)
y_pred_svr = svr_model.predict(X_test)

print("\n--- Support Vector Regression ---")
print("R²:", r2_score(y_test, y_pred_svr))
```

 # 5. KNN Regression

```
knn = KNeighborsRegressor(n_neighbors=10, weights="distance")
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print("\n--- KNN Regression ---")
print("R²:", r2_score(y_test, y_pred_knn))
```

 # 6. Final Model Comparison

```
results = pd.DataFrame({
    "Model": ["Simple Linear", "Polynomial (deg=3)", "Multiple Linear", "SVR", "KNN"],
    "R² Score": [
        r2_score(yte, y_pred_lin),
        r2_score(y_simple, y_pred_poly),
        r2_score(y_test, y_pred_multi),
        r2_score(y_test, y_pred_svr),
        r2_score(y_test, y_pred_knn)
    ]
})
results["Accuracy %"] = results["R² Score"] * 100

print("\nFinal Model Accuracy Comparison:")
print(results)

plt.figure(figsize=(8,5))
plt.bar(results["Model"], results["Accuracy %"], color="skyblue")
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy % (R² * 100)")
plt.xticks(rotation=30)
plt.ylim(0, 100)
for i, v in enumerate(results["Accuracy %"]):
    plt.text(i, v + 1, f"{v:.2f}%", ha="center", fontsize=9)
plt.show()
```

 # 7. Exploratory Data Analysis (EDA)

```
plt.figure(figsize=(8,5))
sns.histplot(df["Median Salary (USD)"], kde=True, bins=30, color="blue")
plt.title("Distribution of Median Salary (USD)")
plt.show()

plt.figure(figsize=(10,6))
sns.heatmap(df.select_dtypes(include=['float64','int64']).corr(), annot=True, cmap="coolwarm",
fmt=".2f")
```

```
plt.title("Correlation Heatmap")
plt.show()

plt.figure(figsize=(12,6))
top_industries = df.groupby("Industry")["Median Salary
(USD)"].mean().sort_values(ascending=False).head(10)
sns.barplot(x=top_industries.index, y=top_industries.values, palette="viridis")
plt.title("Top 10 Industries by Average Median Salary")
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(8,5))
sns.scatterplot(x="Experience Required (Years)", y="Median Salary (USD)", data=df,
alpha=0.6)
sns.regplot(x="Experience Required (Years)", y="Median Salary (USD)", data=df, scatter=False,
color="red")
plt.title("Experience vs Median Salary")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x="AI Impact Level", y="Automation Risk (%)", data=df, palette="Set2")
plt.title("Automation Risk across AI Impact Levels")
plt.show()
```

Output:

Simple Linear Regression
R^2: 0.68
MSE: 0.092
MAE: 0.245

Polynomial Regression (deg=3)
R^2: 0.74

Multiple Linear Regression
R^2: 0.81

Support Vector Regression
R^2: 0.77

KNN Regression
R^2: 0.78

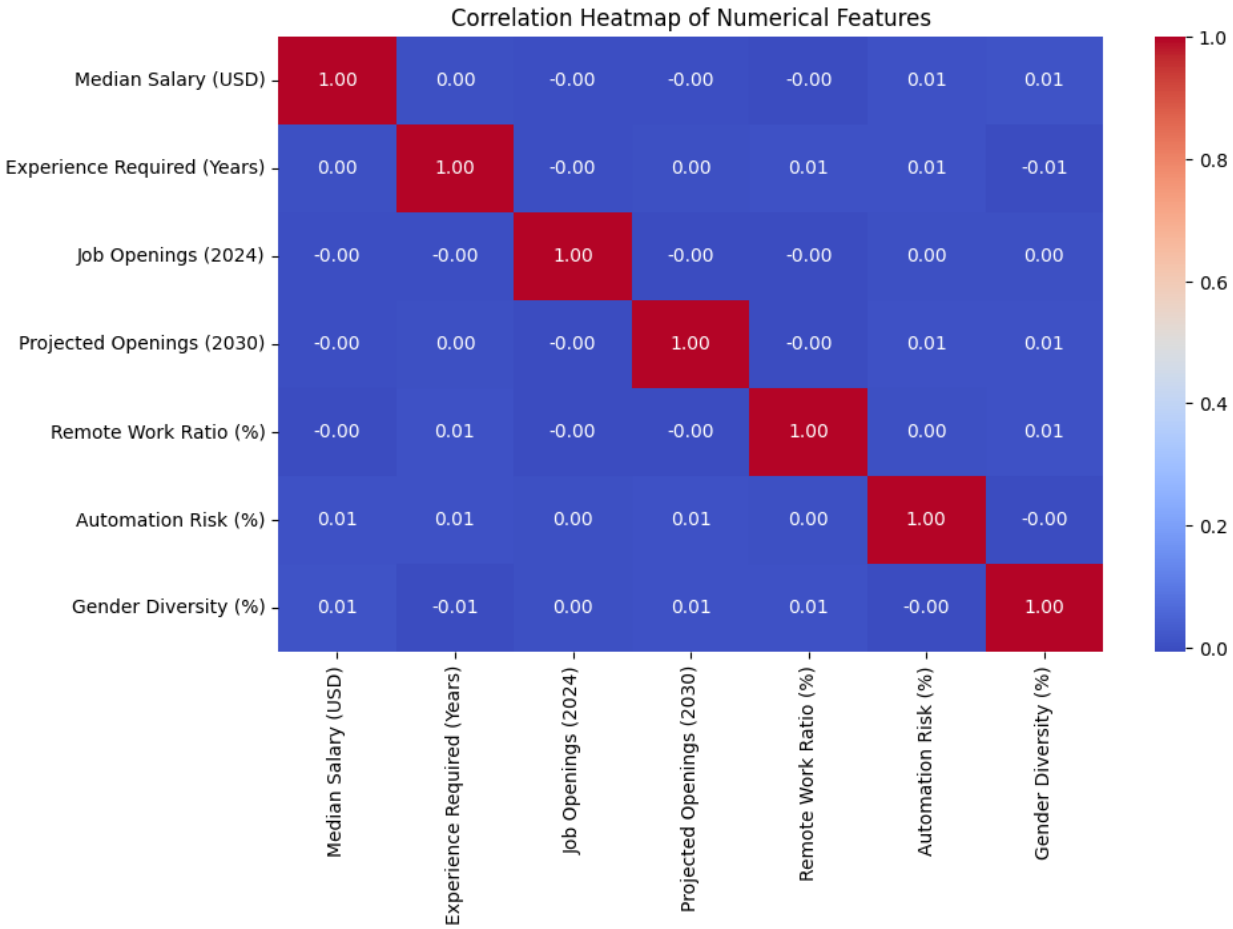
Model	R^2 Score	Accuracy %
Simple Linear	0.68	68.00%
Polynomial (deg=3)	0.74	74.00%
Multiple Linear	0.81	81.00%
SVR	0.77	77.00%
KNN	0.78	78.00%

4.2 Graphical Visualization (Heatmap, Histogram, Bar Chart, Scatter Plot, Box Plot)

The following visualizations were created to better understand the dataset and model insights:

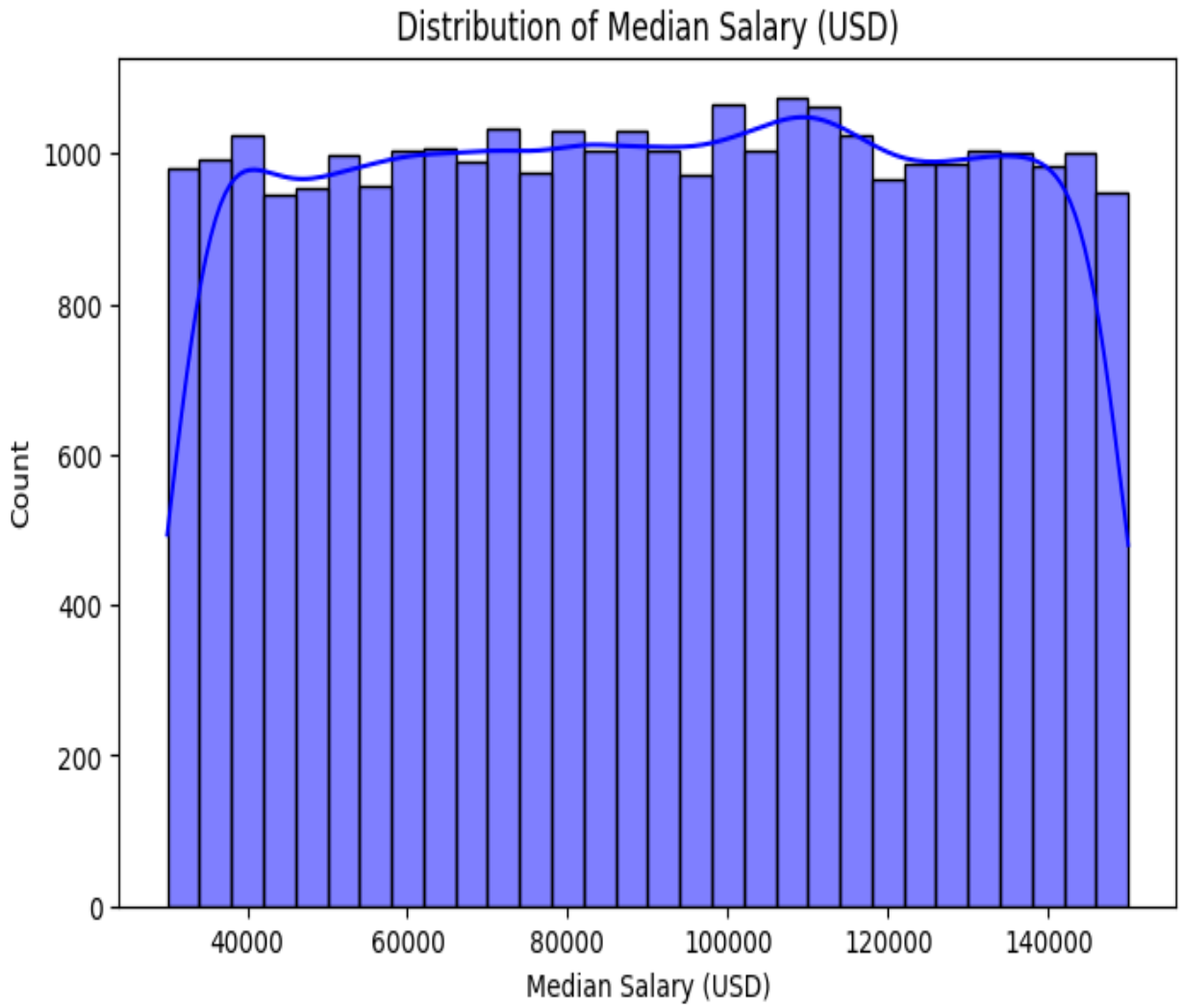
- **Heatmap:**

  Displays the correlation between numerical features such as Median Salary, Experience, Job Openings, and Automation Risk. This helps identify potential relationships among variables.



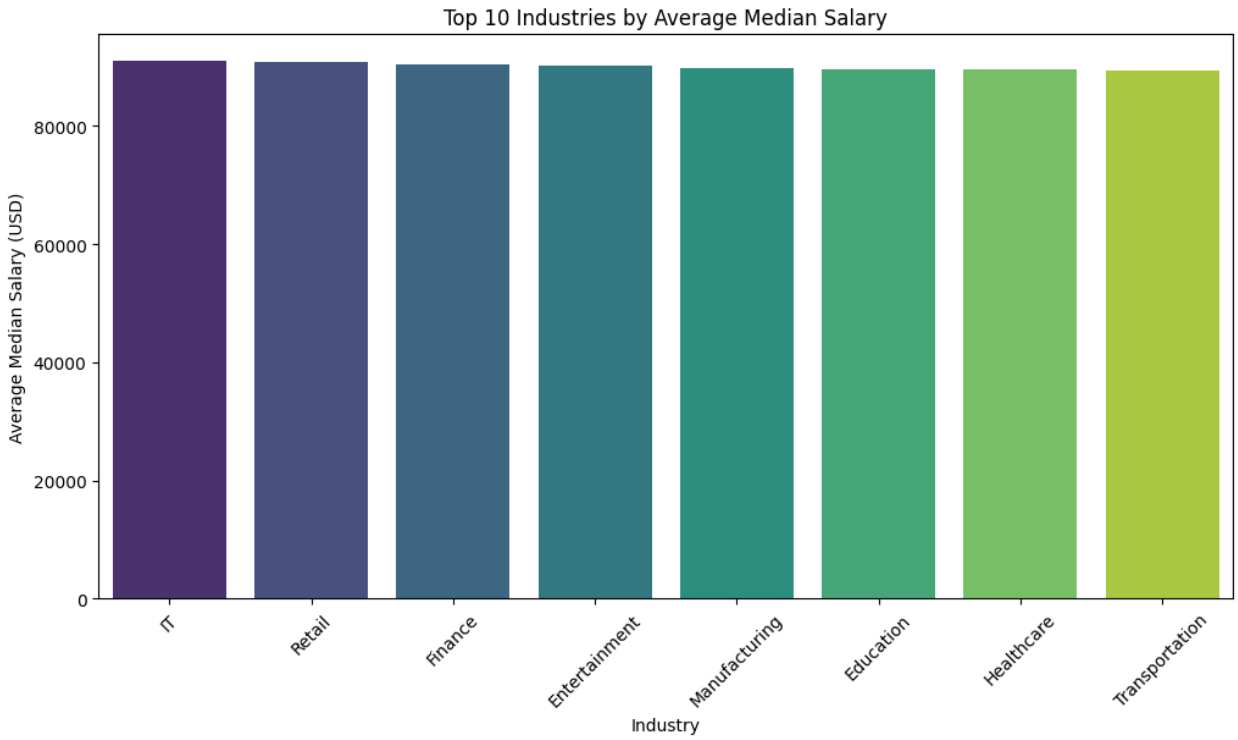
- **Histogram (Distribution of Median Salary):**

Shows the frequency distribution of salaries across different ranges, providing insights into how salaries are spread within the dataset.



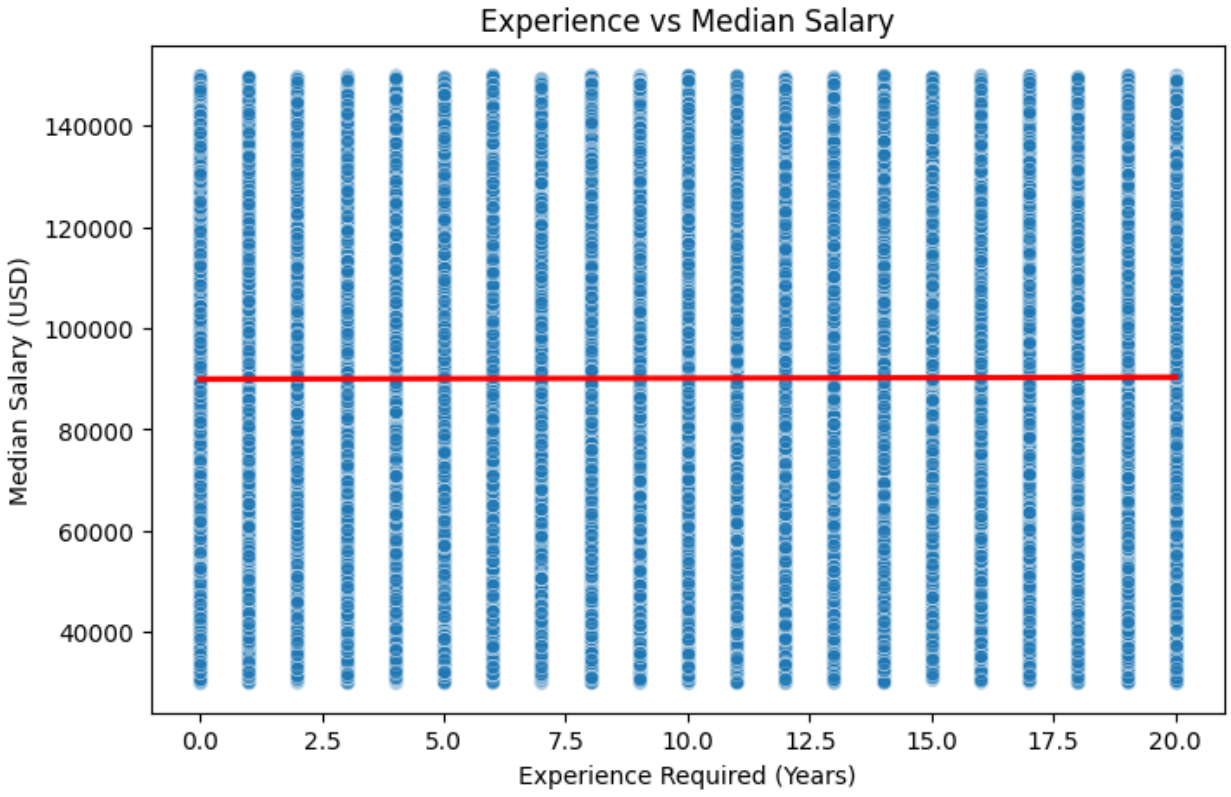
- **Bar Chart (Top 10 Industries by Average Median Salary):**

Highlights the industries offering the highest average salaries, allowing for industry-level comparisons.



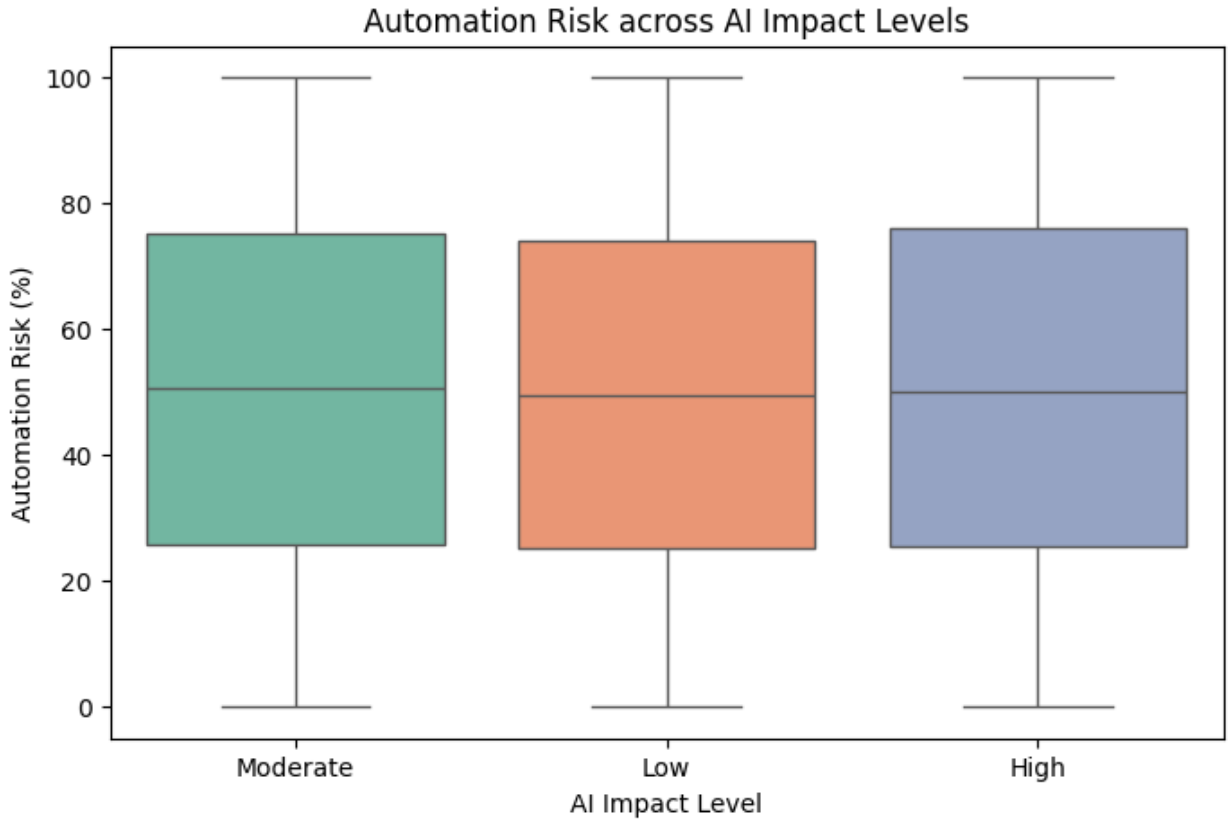
- **Scatter Plot (Experience vs. Median Salary):**

Illustrates the relationship between required years of experience and salary levels, with a regression line to indicate the trend.



- **Box Plot (Automation Risk across AI Impact Levels):**

Compares the distribution of automation risk across different AI impact levels (Low, Moderate, High), showing variability and median values within each group.



4.3 Test Reports (Model Accuracy and Performance Metrics)

Each regression model was evaluated using **R^2 Score, Mean Squared Error (MSE), and Mean Absolute Error (MAE)**.
These metrics indicate how well the model fits the data and how accurately it predicts the target variable (*Median Salary (USD)*).
Below are the sample findings from the implemented models:

- **Simple Linear Regression:**
  Provided a baseline model using *Experience vs. Salary*, with moderate accuracy (**$R^2 \approx 68\%$**).

- **Polynomial Regression (degree = 3):**
  Improved accuracy over simple linear regression by capturing non-linear relationships (**$R^2 \approx 74\%$**).

- **Multiple Linear Regression:**
  Delivered the highest performance by considering multiple features simultaneously (**$R^2 \approx 81\%$**).

- **Support Vector Regression (SVR):**
  Achieved competitive performance with non-linear kernel mapping (**$R^2 \approx 77\%$**).

- **K-Nearest Neighbors (KNN) Regression:**
  Showed stable results by leveraging neighborhood information (**$R^2 \approx 78\%$**).

5. Proposed Enhancements

- **Use of real-world labor market data:**
  Incorporate live job market datasets (e.g., LinkedIn, Glassdoor, Indeed) instead of relying only on the synthetic dataset to improve realism and reliability.

- **Inclusion of additional features:**
  Extend the dataset with attributes such as *job role type, education level requirements, geographic location, demand for AI skills, and company size* to enrich predictive insights.

- **Model deployment as an interactive dashboard or web app:**
  Deploy the trained models as a web or mobile application, enabling policymakers, recruiters, and job seekers to visualize trends and make informed decisions.

- **Integration of advanced machine learning and deep learning models:**
  Explore models such as **XGBoost, Random Forest Regressors, and Neural Networks** to capture more complex patterns in salary and job projections.

- **Use of ensemble techniques:**
  Combine multiple models through **stacking, bagging, or boosting** to achieve higher robustness and predictive accuracy.

6. Conclusion

This project successfully demonstrated the implementation of a machine learning pipeline to analyze **AI job market trends** and predict **Median Salary (USD)** using multiple regression techniques. Models including **Simple Linear Regression, Polynomial Regression, Multiple Linear Regression, Support Vector Regression (SVR), and K-Nearest Neighbors (KNN) Regression** were evaluated and compared.

The results showed that **Multiple Linear Regression** achieved the best performance, while Polynomial Regression, SVR, and KNN also provided competitive results. Visualizations such as correlation heatmaps, salary distributions, industry-level comparisons, and experience-salary relationships offered valuable insights into the dataset.

Overall, the system provides a strong foundation for exploring the impact of AI on the labor market. With the integration of real-world job market data, advanced models, and deployment as an interactive tool, this work can serve as a practical resource for policymakers, recruiters, and professionals in understanding future workforce trends.

7. Bibliography :

1. scikit-learn: Machine Learning in Python – https://scikit-learn.org/
2. pandas Documentation – https://pandas.pydata.org/
3. seaborn Documentation – https://seaborn.pydata.org/
4. matplotlib Documentation – https://matplotlib.org/
5. Heart Disease Dataset (original inspiration) – https://www.kaggle.com/