Link to my GitHub Repo: https://github.com/amitraj8/Neural-FCA-OSDA-HW-

Link to The Dataset:
https://drive.google.com/file/d/1_prpmHVZwT9R2NPEfDzcNOM7egVbmxw0/view?usp=sharing

About the Dataset

I'm currently working with a dataset which contains information about individuals across various locations in India. It focuses on their earnings, which are influenced by several factors, including:

- **Position**: The job title or role of the individual, which may correlate with salary variations.
- **Gender**: Gender distribution of individuals, which can offer insights into gender-based salary disparities.
- **Location**: Different geographic locations in India, reflecting regional salary differences
- **Education**: The highest educational qualification attained by the individual (e.g., B.Tech, BCA, B.Com), which can impact salary levels.
- **Experience in Years**: The number of years of professional experience, which is often a significant factor in determining salary.

The dataset aims to analyze the relationships between these variables and earnings, providing insights into how these factors collectively influence salary distribution across different regions and demographics.

Purpose of this Project

The general purpose of your project is to predict a person's salary based on multiple factors such as Location, Gender, Experience, and Education. Specifically, the goal is to classify whether a person is earning a good salary or not, based on these attributes. This involves analyzing the relationships between these features and salary levels, and leveraging machine learning techniques to make accurate predictions.

Binarization Strategy

In this project, My binarization strategy was to simplify the modelling process and handle categorical data effectively. Several features were selected for binarization based on their characteristics and relevance to the salary prediction task.

- **Education and Position:** I found Position and Education were highly correlated. This correlation is due to the fact that individuals with similar educational backgrounds tend to hold similar positions within their respective fields. Given this relationship, I decided to binarize the **Education** feature rather than the **Position**. This was because binarizing Education was simpler and more straightforward than binarizing Position.

- **Gender:** The **Gender** feature was also binarized into male and female categories.
- **Experience:** The **Experience** feature contains Numerical data varies from 5 to 25 years. I've divided the experience into binary variables representing whether a person has at least 5, 10, 15, or 20 years of experience.
- **Location:** Similarly, the **Location** feature contains categorical values so I binarized each unique location into a separate binary column indicating whether an individual is located in that particular area
- **Salary:** In this project, the target label "Salary" was binarized using a threshold of 1,000,000 INR. I've chosen this threshold as a general benchmark to classify whether a person is earning a "Good Salary" or not, based on the current economic conditions and salary trends in India. If a person's salary exceeds this threshold, they are classified as earning a good salary (1), while those earning below it are classified as earning a bad salary (0).

Setup

In this step, I've created a new dataset that contains all the binarized features essential for predicting whether a person earns a good salary or not. The original dataset, which includes various categorical features such as Gender, Education, Position, and Location, was transformed into binary format to make it suitable for machine learning models.

Features	**Binarized_Features**
Education	6 features, eg: Education_B.B.A/ B.M.S, Education_B.Com, Education_B.Sc etcs
Gender	2 Features, Eg: Male & Female
Experience	4 features Eg; Experience_>=5 Experience_>=10 Experience_>=15 Experience_>=20
Location	24 features Eg; Location_Gurugram Location_Mumbai Location_New Delhi
Salary	2 features Eg; Good_Salary Bad_Salary

Neural FCA Model

Implementation:

In this step, I’ve implemented a **Neural Formal Concept Analysis (Neural FCA)** model to predict whether a person earns a good salary based on transformed features such as Gender, Education, Location, and Experience. The process involved applying **Formal Concept Analysis (FCA)** to identify formal concepts, which are pairs of objects and attributes that share relationships within the data. These FCA concepts were then integrated into a neural network model, enriching it with additional insights from the data. The neural network was trained to classify salary as either "Good" or "Bad" based on these features, and FCA concepts helped guide the model by highlighting important relationships between the input attributes. This hybrid approach combined the interpretability of FCA with the predictive power of neural networks.

Results:

The performance of the Neural FCA model yielded the following evaluation metrics: **Accuracy: 0.43**, **Precision: 0.86**, **Recall: 0.46**, and **F1-score: 0.60**. These results suggest that while the model performs relatively well in identifying "Good Salary" cases, its ability to recall all true positives (individuals with good salaries) could be improved. The high precision indicates that when the model predicts someone as earning a good salary, it is correct most of the time, but the lower recall suggests that the model misses some true positive cases. The F1-score of 0.60 reflects a reasonable balance between precision and recall, showing that the Neural FCA model is effective in predicting salary categories, though there is room for improvement in capturing all relevant cases.

Decision Tree

Implementing the Decision Tree Model

In this step, I used **Decision Tree Classifier** to predict whether a person is earning a good salary based on the transformed features, such as Gender, Education, Location, and Experience. The Decision Tree algorithm works by recursively splitting the dataset into subsets based on feature values, creating a tree-like structure where each internal node represents a feature (or attribute), and each leaf node represents a predicted class (in this case, whether the salary is good or bad). The model was trained on the binarized dataset, where the features and target label were transformed for optimal performance. Decision Trees are interpretable models, making them particularly useful for understanding how different features influence the classification of salary levels.

Results

The Decision Tree model achieved the following results: **Accuracy: 0.64**, **Precision: 1.0**, **Recall: 0.62**, and **F1-score: 0.76**. These results show that the model performs quite well, particularly in terms of

precision, where it correctly predicted all "Good Salary" cases without any false positives. However, while precision is perfect, the **recall** value of 0.62 indicates that the model still misses some "Good Salary" cases, suggesting it is not capturing all potential true positives. The **F1-score** of 0.76 reflects a strong balance between precision and recall, demonstrating that the Decision Tree classifier is effective at predicting salary categories. Overall, the Decision Tree model provides better performance than the Neural FCA model, with higher accuracy and F1-score, though there is still room for further improvement in recall.

Random Forest

Implementation:

In this step, I used **Random Forest Classifier** to predict whether a person is earning a good salary based on the binarized features like Gender, Education, Location, and Experience. A Random Forest is an ensemble learning method that combines multiple Decision Trees to improve performance and reduce the risk of overfitting. Each tree is trained on a random subset of the data, and the final prediction is made by averaging the results (for regression) or taking a majority vote (for classification) across all the trees in the forest. By utilizing this ensemble approach, Random Forest models tend to be more robust than individual Decision Trees, especially when dealing with noisy data or complex relationships.

Results:

The Random Forest model produced the following results: **Accuracy: 0.71**, **Precision: 1.0**, **Recall: 0.69**, and **F1-score: 0.82**. These results show that the Random Forest model significantly outperforms the Decision Tree model in terms of **accuracy**, achieving a higher rate of correct predictions overall. Like the Decision Tree model, the Random Forest model achieved **perfect precision**, indicating that all predictions of "Good Salary" were correct without any false positives. The **recall** value of 0.69 demonstrates that the model did a better job of identifying actual "Good Salary" cases compared to the Decision Tree, though there is still room for improvement in capturing all true positives. The **F1-score** of 0.82 reflects an even better balance between precision and recall, making it the best-performing model so far. Overall, the Random Forest Classifier provides strong performance, offering better generalization and accuracy than both the Neural FCA and Decision Tree models.

Naive Bayes Model

Implementation:

In this step, I used **Naive Bayes Classifier** for predicting whether a person is earning a good salary based on the available features. Naive Bayes is a probabilistic classifier based on Bayes' Theorem, which assumes that the features are conditionally independent given the class. While this assumption of independence may not always hold in real-world data, Naive Bayes can still perform well in many cases, especially when dealing with high-dimensional data or when the features are loosely correlated. This model was trained on the binarized features of **Gender**, **Education**, **Location**, and **Experience**, and used the class labels to calculate probabilities for each class.

Results:

The Naive Bayes model produced the following results: **Accuracy: 0.29**, **Precision: 1.0**, **Recall: 0.23**, and **F1-score: 0.38**. While the **precision** is perfect (1.0), indicating that all predictions of "Good Salary" were correct, the model shows a significant **imbalance** in performance, with a very low **recall** of 0.23. This means that the model was only able to identify about 23% of all actual "Good Salary" cases, suggesting it is heavily biased toward predicting the negative class ("Bad Salary"). The **accuracy** of 0.29 is quite low, reflecting the model's inability to capture most of the true positives. The **F1-score** of 0.38 further indicates that while the model is very precise in its predictions, it is not good at identifying the "Good Salary" instances in the data. Overall, the Naive Bayes model underperformed in this context, likely due to its strong assumption of feature independence, which may not be realistic for the given dataset.

Logistic Regression

Implementation:

In this step, I've used **Logistic Regression Classifier** to predict whether a person is earning a good salary based on features such as **Gender**, **Education**, **Location**, and **Experience**. Logistic Regression is a linear model for binary classification that uses a logistic function to model the probability of the positive class. The model was trained using the binarized features, and it learned the relationship between these features and the target label ("Good Salary" or "Bad Salary"). Logistic Regression is a popular choice for binary classification tasks because it provides probabilities for class predictions and is interpretable.

Results:

The Logistic Regression model showed excellent performance with the following results: **Accuracy: 0.86**, **Precision: 1.0**, **Recall: 0.85**, and **F1-score: 0.92**. The **precision** of 1.0 indicates that every time the model predicted "Good Salary," it was correct. The **recall** of 0.85 means the model correctly identified 85% of the true "Good Salary" cases. The high **F1-score** of 0.92 reflects a strong balance between precision and recall, making the model very effective at both identifying and accurately predicting "Good Salary" instances. The **accuracy** of 0.86 further confirms that the Logistic Regression model is performing well overall. This model achieved strong results, suggesting it is well-suited for this classification task, especially in comparison to other models tested.

KNN Model

Implementation:

In this step, I’ve used the **K-Nearest Neighbors (KNN) Classifier** to predict whether a person is earning a good salary based on the given features such as **Gender**, **Education**, **Location**, and **Experience**. KNN is a non-parametric, instance-based learning algorithm that classifies a data point based on the majority class among its k-nearest neighbors in the feature space. The model does not make any assumptions about the underlying data distribution, which can be beneficial for real-world, complex datasets like ours. We used a variety of distances to measure similarity between data points, and the optimal value of "k" was chosen based on cross-validation.

Results:

The **KNN model** achieved the following results: **Accuracy: 0.79**, **Precision: 0.92**, **Recall: 0.85**, and **F1-score: 0.88**. The **precision** of 0.92 indicates that when the model predicted "Good Salary," it was correct approximately 92% of the time. The **recall** of 0.85 means that the model correctly identified 85% of the true "Good Salary" cases. With an **F1-score** of 0.88, the model strikes a strong balance between precision and recall, providing a good overall measure of its effectiveness. The **accuracy** of 0.79 further demonstrates the model's solid performance. These results suggest that the KNN classifier performed well and is capable of providing accurate predictions for salary classification, showing its potential as a reliable model for this task.

Conclusion:

In this project, I’ve developed a machine learning pipeline to predict whether a person in India is earning a good salary or not using various features such as location, gender, experience, and education. Through feature binarization, I transformed categorical variables into binary formats to make them suitable for machine learning models. I’ve applied several classification algorithms including Neural FCA, Decision Trees, Random Forests, Naive Bayes, Logistic Regression, and K-Nearest Neighbors (KNN) to analyze the impact of these features on salary prediction.

The results showed that different models provided varying levels of performance, with **Logistic Regression** achieving the highest accuracy and F1-score (85.7% and 91.7%, respectively). While other models such as Decision Trees and Random Forests also performed well, Naive Bayes had significantly lower performance. Overall, the project demonstrates the effectiveness of machine learning techniques in predicting salary outcomes based on demographic and professional factors, and highlights the importance of selecting the right model for optimal performance in real-world applications.

Link to my GitHub Repo: https://github.com/amitraj8/Neural-FCA-OSDA-HW-

Link to The Dataset:
https://drive.google.com/file/d/1_prpmHVZwT9R2NPEfDzcNOM7egVbmxw0/view?usp=sharing