CAPSTONE PROJECT

EMPLOYEE INCOME ANALYSIS ML PROJECT

Presented By:
1. Chandan Pandey-Shri Lal Bahadur Shastri Degree College-Department of Technology

edunet foundation

OUTLINE

- **Problem Statement**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment (Step by Step Procedure)**
- **Result**
- **Conclusion**
- **References**

PROBLEM STATEMENT

- This project is focused on building a machine learning model to predict whether an employee's income is greater than 50,000 or less than or equal to 50,000 per year. The prediction is based on a set of demographic and employment-related features from the "adult 3.csv" dataset. The key challenge is to preprocess the dataset, which contains mixed data types and missing values, and then train various classification models to find the one with the highest accuracy. The ultimate goal is to create a reliable predictive tool for salary level classification

SYSTEM APPROACH

- **System requirements**
  - **Language:** Python 3
  - **Environment:** Jupyter Notebook, .venv
  - **Deployment:** Streamlit
  - **RAM:** Minimum 4GB
- **Library required to build the model**
  - Pandas
  - matplotlib.pyplot
  - scikit-learn
  - Joblib
  - streamlit:

ALGORITHM & DEPLOYMENT

1. Data Loading and Exploration:

- The dataset ‘adult 3.csv’ was loaded into a pandas DataFrame. Initial exploration was done using .head(), .shape, and .isna().sum() to understand its structure and check for null values.

2. Data Cleaning:

- Missing values, represented by '?', in the workclass and occupation columns were identified and replaced with the category 'Others’.
- Rare categories like 'Without-pay' and 'Never-worked' were filtered out from the dataset.

ALGORITHM & DEPLOYMENT

3. Outlier Treatment:

- Box plots were used to visualize outliers in numerical columns like age, educational-num, and hours-per-week. The data was then filtered to remove these outliers, resulting in a cleaner dataset.

4. Feature Engineering:

- The education column was dropped as it was redundant with the educational-num column.

5. Data Transformation:

- Categorical columns (workclass, marital-status, occupation, relationship, race, gender, native-country) were converted into numerical format using LabelEncoder.

ALGORITHM & DEPLOYMENT

6. Model Training and Evaluation:

- The data was split into training (80%) and testing (20%) sets.
- Five different classification models were trained and evaluated: Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gradient Boosting.
- The accuracy and a detailed classification report were generated for each model.

7. Model Selection and Saving:

- The Gradient Boosting Classifier was identified as the best-performing model based on its accuracy score.
- This model was saved as a best_model.pkl file using joblib.

8. Deployment:

- A web application script (app.py) was created using Streamlit to load the saved model and allow for both single and batch predictions.

RESULT

- Screen snaps of the code and output:

```
1  data=data[(data['age']<=75)&(data['age']>=17)]
```
✓ 0.0s

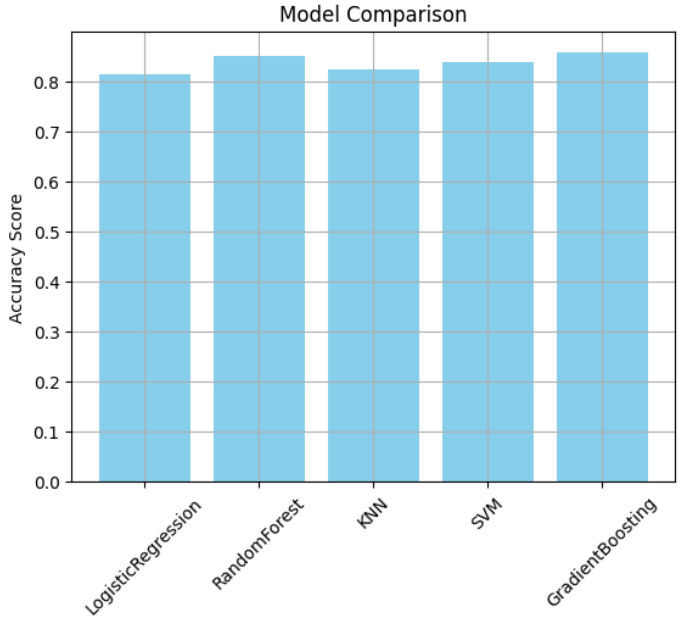
```
1  x=data.drop(columns=['income'])
2  y=data['income']
3  x
```
✓ 0.1s

```
Click to add a breakpoint
eline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, OneHotEncoder

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

models = {
    "LogisticRegression": LogisticRegression(),
    "RandomForest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
    "GradientBoosting": GradientBoostingClassifier()
}

results = {}
```




```
LogisticRegression: 0.7968
RandomForest: 0.8502
KNN: 0.7704
SVM: 0.7884
GradientBoosting: 0.8571

✅ Best model: GradientBoosting with accuracy 0.8571
✅ Saved best model as best_model.pkl
```

edunet foundation

RESULT

- Github link:
  https://github.com/chandanpandeys/IBMInternshipProjects/

CONCLUSION

- The project successfully developed a robust pipeline for predicting employee salary levels. After comprehensive data cleaning, outlier removal, and feature engineering, five different classification models were compared.
- **The Gradient Boosting Classifier** emerged as the most effective model, achieving an accuracy of **85.71%** on the test data. This demonstrates its strong capability in classifying salaries based on the given features. The final model was saved and prepared for deployment in a Streamlit web application, providing a practical tool for salary prediction.

REFERENCES

1. Pandas documentation: https://pandas.pydata.org/
2. Matplotlib documentation: https://matplotlib.org/stable/
3. Scikit-learn documentation: https://scikit-learn.org/stable/
4. Predicting Salary Expectations with a Supervised Machine Learning Regression Model (Medium): https://medium.com/inst414-data-science-tech/predicting-salary-expectations-with-a-supervised-machine-learning-regression-model-5a59bf552265
5. Employee-Salary-Prediction-and-Optimization-Using-Predictive-Analytics: https://github.com/ChetanBhangare/Employee-Salary-Prediction-and-Optimization-Using-Predictive-Analytics
6. Salary Prediction Using Machine Learning: ijsdr.org/papers/IJSDR2404019.pdf

THANK YOU

edunet foundation