
CAPSTONE PROJECT

"EMPLOYEE SALARY PREDICTION USING PAYPREDICTOR" *AN EMPIRICAL STUDY OF MACHINE LEARNING ALGORITHMS*

Presented By:

**VIDYADHEESHA M PANDURANGI
ADHIYAMAAN COLLEGE OF ENGINEERING
B.E. ELECTRONICS AND COMMUNICATION ENGINEERING**

OUTLINE

- **Problem Statement**
- **System Development Approach**
- **Algorithm & Deployment**
- **Result**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

Income disparity remains a pervasive and complex societal challenge, with far-reaching implications for individuals and broader economic stability. The ability to accurately predict an individual's income level based on readily available demographic and employment data holds significant value across various domains. For instance, financial institutions could leverage such predictions for more nuanced credit risk assessments, tailoring financial products to specific income brackets. Policy makers, on the other hand, might utilize these insights to identify at-risk populations, inform social welfare programs, or analyze the potential impact of economic policies. Businesses could also benefit by optimizing marketing strategies and product development for different income segments, fostering more targeted and effective outreach.

The current landscape, however, often grapples with the inherent complexity of socioeconomic factors influencing income. Traditional analytical approaches may struggle to capture the intricate, non-linear relationships between variables such as age, workclass, education, marital status, occupation, and geographical origin, alongside financial indicators like capital gains and losses, and weekly working hours. This leads to less precise predictions, potentially resulting in misallocated resources, inaccurate risk profiling, or less effective policy interventions.

This project directly addresses this challenge by developing a robust predictive model capable of categorizing individual income into distinct brackets (specifically, ">50K" or "<=50K"). Our aim is to move beyond simplistic correlations and explore advanced machine learning techniques to unveil the subtle patterns within a comprehensive dataset. By doing so, we intend to provide a more reliable and actionable tool for understanding and forecasting income levels, ultimately contributing to more informed decision-making across financial, governmental, and commercial sectors.

SYSTEM APPROACH

The System Approach section outlines the overall strategy, development methodology, system requirements, and libraries required for building and deploying the PayPredictor application. It ensures a structured approach toward the project lifecycle – from data acquisition and preprocessing to model development, evaluation, and deployment via a web-based interface.

System Requirements

To build and execute this machine learning web application efficiently, the following hardware and software specifications were required:

► Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i3 or equivalent
RAM	4 GB (8 GB recommended)
Storage	1 GB free space
Display	1024 x 768 resolution or higher

SYSTEM APPROACH

► Software Requirements

Software Component

Operating System

Programming Language

Python IDE

Web Framework

Browser

Specification

Windows 10 / Linux / macOS

Python 3.7+

VS Code / Jupyter Notebook / PyCharm

Streamlit

Chrome / Firefox / Edge

SYSTEM APPROACH

Required Libraries to Build the Model

The model leverages the Python ecosystem's powerful libraries for **data handling, preprocessing, modeling, and deployment**. Below is a list of essential libraries used and their purposes:

Library	Purpose
pandas	Data loading, manipulation, and DataFrame creation
numpy	Numerical computations and array manipulation
scikit-learn	Building the preprocessing pipeline and machine learning model
joblib	Saving and loading models and pipelines
Streamlit	Creating the interactive and responsive web-based user interface
warnings	Suppressing unnecessary warning messages

ALGORITHM & DEPLOYMENT

Dataset Description

The dataset contains various attributes like:

- Age
- Education
- Occupation
- Workclass
- Marital Status
- Relationship
- Hours-per-week
- Gender
- Native Country

Target Variable:

- Income (either $\leq 50K$ or $> 50K$)

ALGORITHM & DEPLOYMENT

Step 1: Data Collection

The dataset (Dataset.csv) is loaded using Pandas:

```
import pandas as pd  
data = pd.read_csv("Dataset.csv")
```

Step 2: Handling Outliers

To visualize and optionally remove outliers, boxplots were generated for features like age and hours-per-week.

```
import matplotlib.pyplot as plt  
plt.boxplot(x=data['age'])  
plt.title("Boxplot for Age (Outlier Detection)")  
plt.show()
```

This helps understand the distribution and presence of outliers, but no explicit outlier removal was done in code—possibly because tree-based algorithms (like ExtraTrees and CatBoost) handle them better.

ALGORITHM & DEPLOYMENT

Step 3: Feature and Target Split

The dataset is divided into features x and target variable y .

```
x = data.drop(columns = ["income"])  
y = data["income"]
```

Step 4: Preprocessing Using ColumnTransformer

Numerical features are scaled using StandardScaler, while categorical features are encoded using OneHotEncoder. The ColumnTransformer helps apply these transformations selectively.

```
from sklearn.compose import ColumnTransformer  
  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
  
numeric_features = ['age', 'hours-per-week']  
  
categorical_features = ['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'gender',  
                        'native-country']  
  
preprocessor = ColumnTransformer(transformers=[ ('num', StandardScaler(),  
numeric_features), ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)  
])
```

ALGORITHM & DEPLOYMENT

Step 5: Train-Test Split

The dataset is split into training and testing sets using an 80-20 ratio:

```
from sklearn.model_selection import train_test_split  
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=43)
```

Step 6: Data Cleaning

Certain columns that were either not beneficial or had high correlation with others were dropped and renamed for consistency, null values (?) replaced with “Not Specified”, and cleaned:

```
data = data.drop(columns=['fnlwgt', 'educational-num', 'capital-gain', 'capital-loss'])  
data = data.replace("?", "Others", inplace=True)
```

ALGORITHM & DEPLOYMENT

Algorithms Used and Justification

The following classification algorithms were tested and evaluated for model performance:

1. CatBoost Classifier:

- **Type:** Gradient Boosting on Decision Trees
- **Advantage:** Handles categorical variables automatically.
- **Use Case:** High-performance classification problems.

Code Snippet:

```
from catboost import CatBoostClassifier  
  
model = CatBoostClassifier(verbose=0)
```

ALGORITHM & DEPLOYMENT

2. LightGBM Classifier:

- **Type:** Gradient Boosting Framework (leaf-wise)
- **Advantage:** Faster than other boosting methods.
- **Use Case:** Suitable for large datasets.

Code Snippet:

```
import lightgbm as lgb  
  
model = lgb.LGBMClassifier()
```

3. Quadratic Discriminant Analysis (QDA)

- **Type:** Probabilistic Classifier
- **Advantage:** Captures non-linear decision boundaries.
- **Limitation:** Sensitive to multicollinearity and outliers.

ALGORITHM & DEPLOYMENT

Code Snippet:

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis  
  
model = QuadraticDiscriminantAnalysis()
```

4. Extra Trees Classifier

- **Type:** Ensemble Learning, Random Forest variant
- **Advantage:** High accuracy, faster than RF.
- **Use Case:** Feature importance analysis and stable predictions.

Code Snippet:

```
from sklearn.ensemble import ExtraTreesClassifier  
  
model = ExtraTreesClassifier()
```

ALGORITHM & DEPLOYMENT

Step 7: Model Evaluation and Selection

Each model is trained using a pipeline that integrates preprocessing and model fitting:

```
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report
results, pipelines = {}
for name, model in models.items():
    pipe = Pipeline([("preprocessor", preprocessor), ("model", model)])
    pipe.fit(xtrain, ytrain)
    y_pred = pipe.predict(xtest)
    acc = accuracy_score(ytest, y_pred)
    results[name] = acc
    pipelines[name] = pipe
    print(f'{name} Accuracy: {acc:.4f}')
    print(classification_report(ytest, y_pred))
```

ALGORITHM & DEPLOYMENT

Step 7: Model Evaluation and Selection

The model with the highest accuracy is selected as the final model.

```
from sklearn.pipeline import Pipeline  
  
import joblib  
  
best_model_name = max(results, key=results.get)  
  
best_pipeline = pipelines[best_model_name]  
  
joblib.dump(best_pipeline, "best_model.pkl")
```

Step 8: Web App Deployment using Streamlit

A Streamlit app was built to collect inputs and predict salary class.

Key UI Elements:

- Sliders, dropdowns, and radio buttons to collect input features.
- A centered prediction button.
- Output display for prediction result.

ALGORITHM & DEPLOYMENT

SAMPLE CODE SNIPPET:

```
import streamlit as st
import pandas as pd
import joblib
model = joblib.load("best_model.pkl")
age = st.slider("Age", 18, 90, 30)
education = st.selectbox("Education", [...])....
input_df = pd.DataFrame([ {...} ])
if st.button("Predict Income Category"):
    prediction = model.predict(input_df)[0]
    st.success(f"Predicted Income Category: **{prediction}**")
```

Step 10: Launching the App

To launch the Streamlit app locally:

```
streamlit run app.py
```


ALGORITHM & DEPLOYMENT

Step 11: Deployment Using Streamlit Cloud

To make my application accessible to users globally, I deployed it using Streamlit Cloud.

Files I Prepared:

PayPredictor.py: This contains the complete Streamlit app code.

best_model.pkl: The serialized pipeline containing both the preprocessing steps and the trained best model.

Dataset.csv: It contains the dataset on which the model has been trained.

requirements.txt: A file listing all the dependencies required to run the app.

Code_for_the_project: Notebook environment where data preprocessing, cleaning and training of the model was done.

streamlit

pandas

scikit-learn

catboost

lightgbm

joblib

Matplotlib

ALGORITHM & DEPLOYMENT

Steps for Deployment:

- Pushed all the project files into a GitHub repository. (*PayPredictor-Employee-Salary-Prediction*)
- I visited <https://share.streamlit.io> and signed in using my GitHub account.
- I clicked on “Create app”.
- I selected the GitHub repository (PayPredictor) containing my project, then chose:
 - **Branch:** main
 - **App file:** PayPredictor.py
- I clicked on the "Deploy" button and waited for a few moments.
- My application was then deployed successfully and could be accessed through a public URL like:
<https://paypredictorai.streamlit.app/>

RESULT

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K
...
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

48842 rows × 15 columns

Figure: 1 – Preview of the dataset

RESULT

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K

Figure: 2 – Head of the dataset

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

Figure: 3 – Tail of the dataset

RESULT

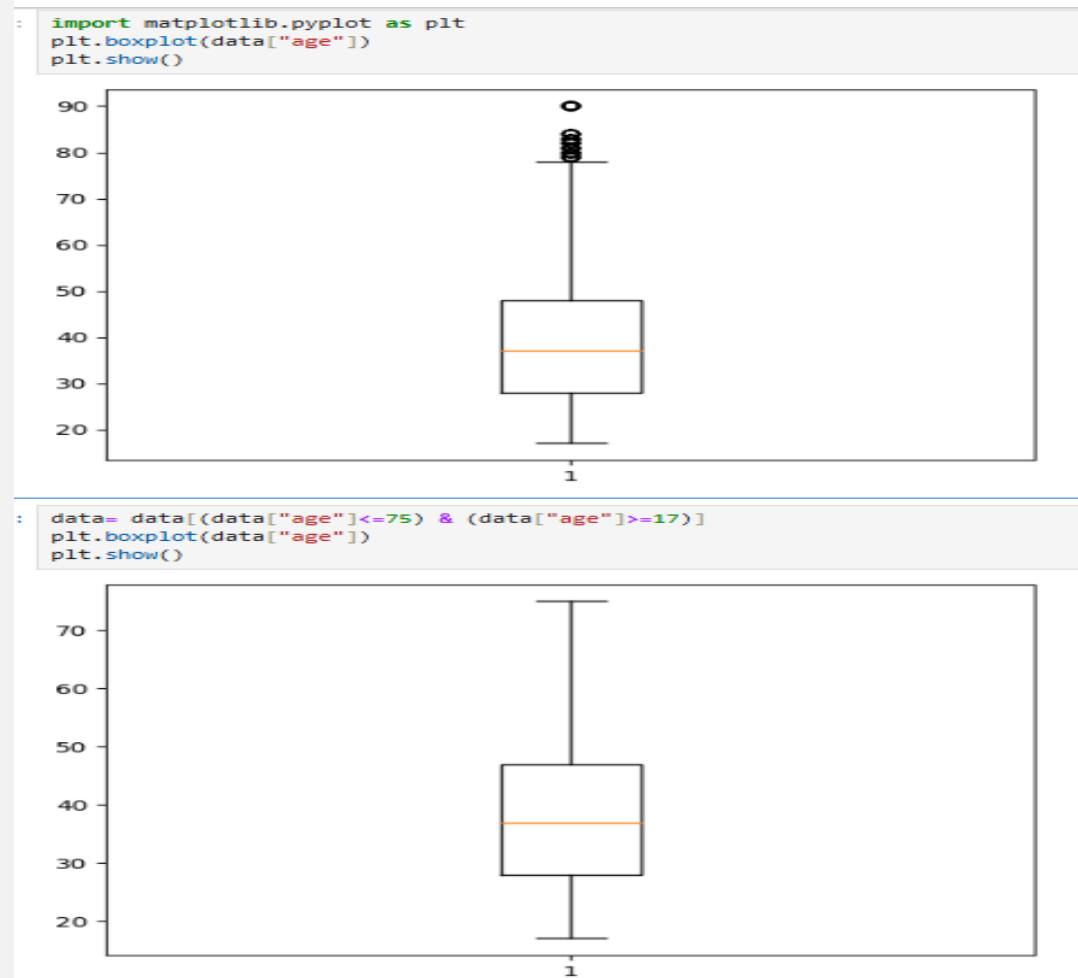


Figure: 4 – Removal of outliers using matplotlib(Boxplot Method) from the dataset

RESULT

```
CatBoost Accuracy: 0.8446
precision    recall  f1-score   support

<=50K      0.85    0.96    0.90    7370
>50K       0.81    0.48    0.60    2399

accuracy    0.84    0.84    0.84    9769
macro avg   0.83    0.72    0.75    9769
weighted avg 0.84    0.84    0.83    9769

[LightGBM] [Info] Number of positive: 9288, number of negative: 29785
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000686 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 623
[LightGBM] [Info] Number of data points in the train set: 39073, number of used features: 6
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.237709 -> initscore=-1.165282
[LightGBM] [Info] Start training from score -1.165282
LightGBM Accuracy: 0.8444
precision    recall  f1-score   support

<=50K      0.85    0.96    0.90    7370
>50K       0.81    0.48    0.60    2399

accuracy    0.84    0.84    0.84    9769
macro avg   0.83    0.72    0.75    9769
weighted avg 0.84    0.84    0.83    9769

QDA Accuracy: 0.7940
precision    recall  f1-score   support

<=50K      0.81    0.95    0.87    7370
>50K       0.68    0.31    0.42    2399

accuracy    0.79    0.79    0.79    9769
macro avg   0.74    0.63    0.65    9769
weighted avg 0.78    0.79    0.76    9769

Extra Trees Accuracy: 0.7921
precision    recall  f1-score   support

<=50K      0.85    0.88    0.86    7370
>50K       0.59    0.52    0.55    2399

accuracy    0.79    0.79    0.79    9769
macro avg   0.72    0.70    0.71    9769
weighted avg 0.78    0.79    0.79    9769
```

Figure: 5 – Accuracy score and Classification Report results of different machine learning algorithms based on the dataset

RESULT

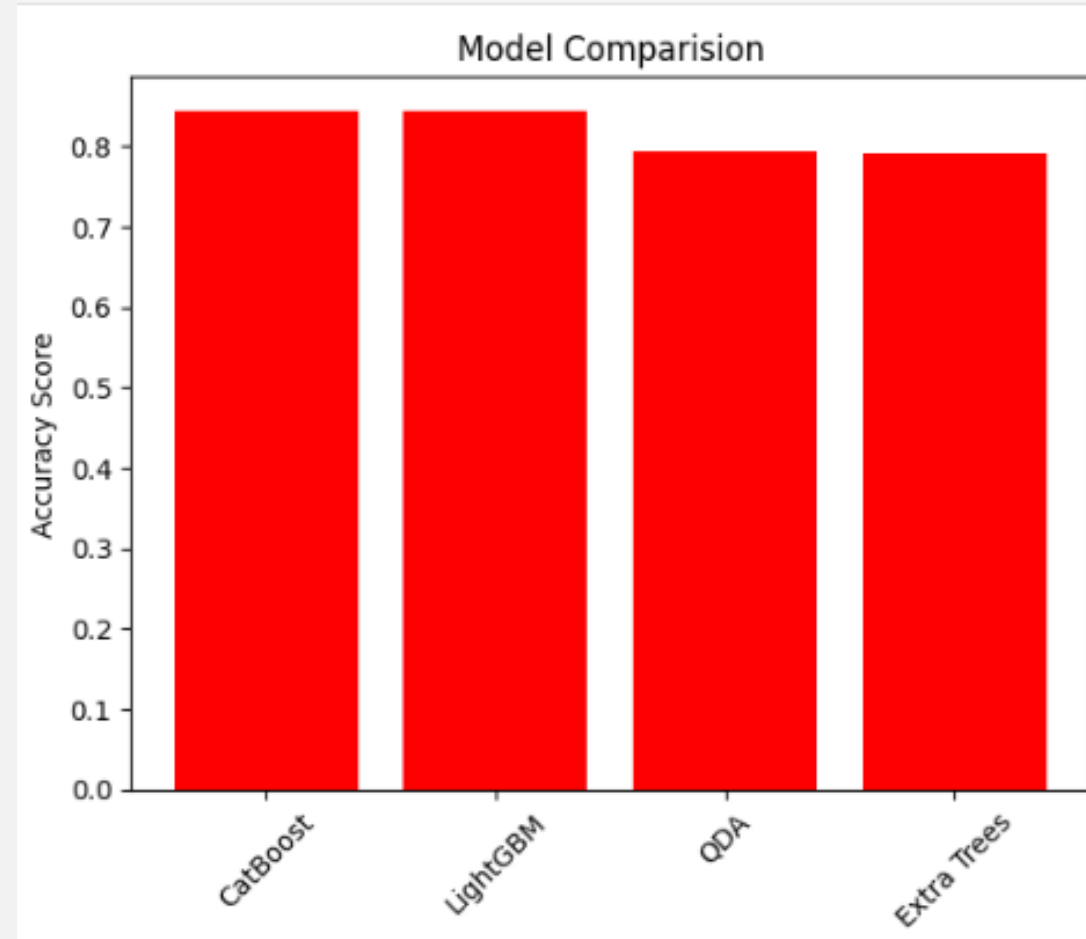
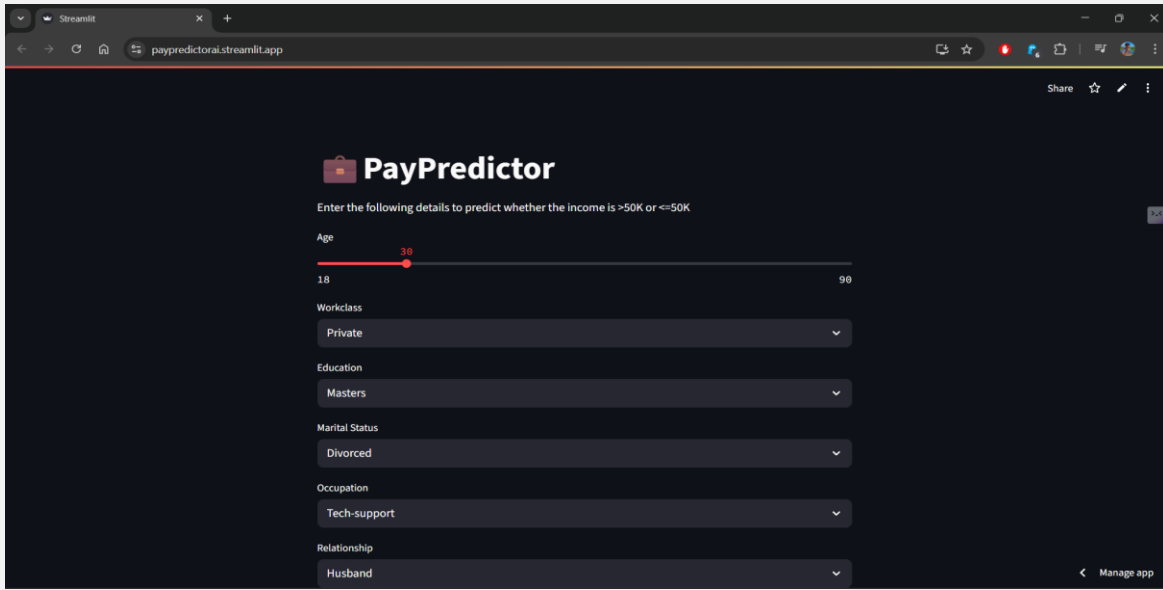


Figure: 6 – Comparison graph representing the accuracy score of different machine learning algorithms

RESULT



PayPredictor

Enter the following details to predict whether the income is >50K or <=50K

Age: 30 (range 18 to 90)

Workclass: Private

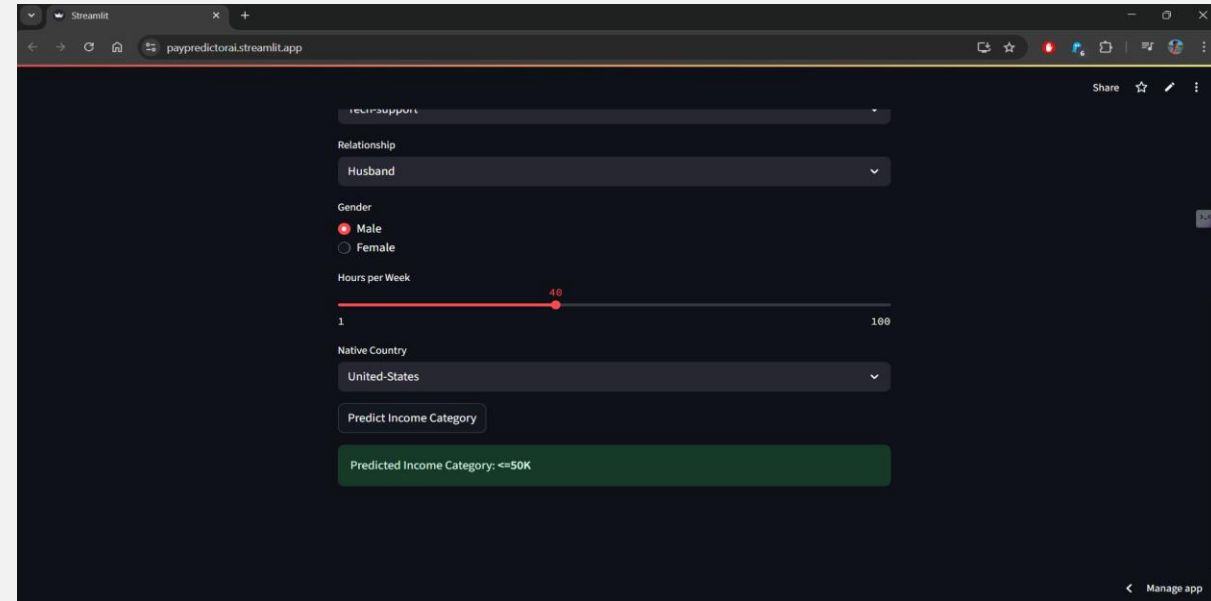
Education: Masters

Marital Status: Divorced

Occupation: Tech-support

Relationship: Husband

Manage app



Relationship: Husband

Gender: ☒ Male ☐ Female

Hours per Week: 40 (range 1 to 100)

Native Country: United-States

Predict Income Category

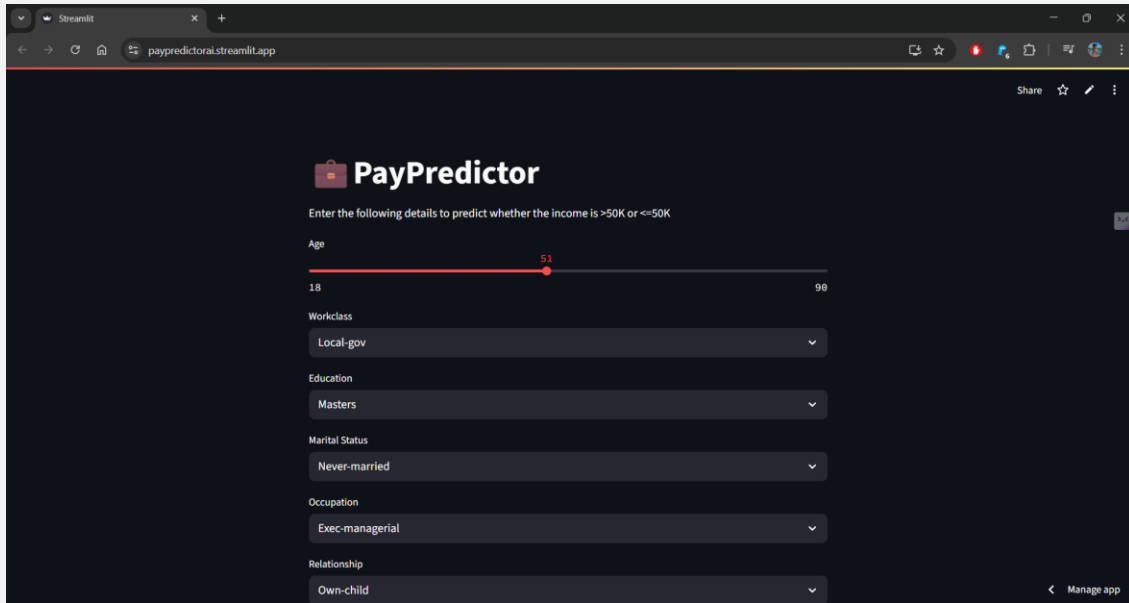
Predicted Income Category: <=50K

Manage app

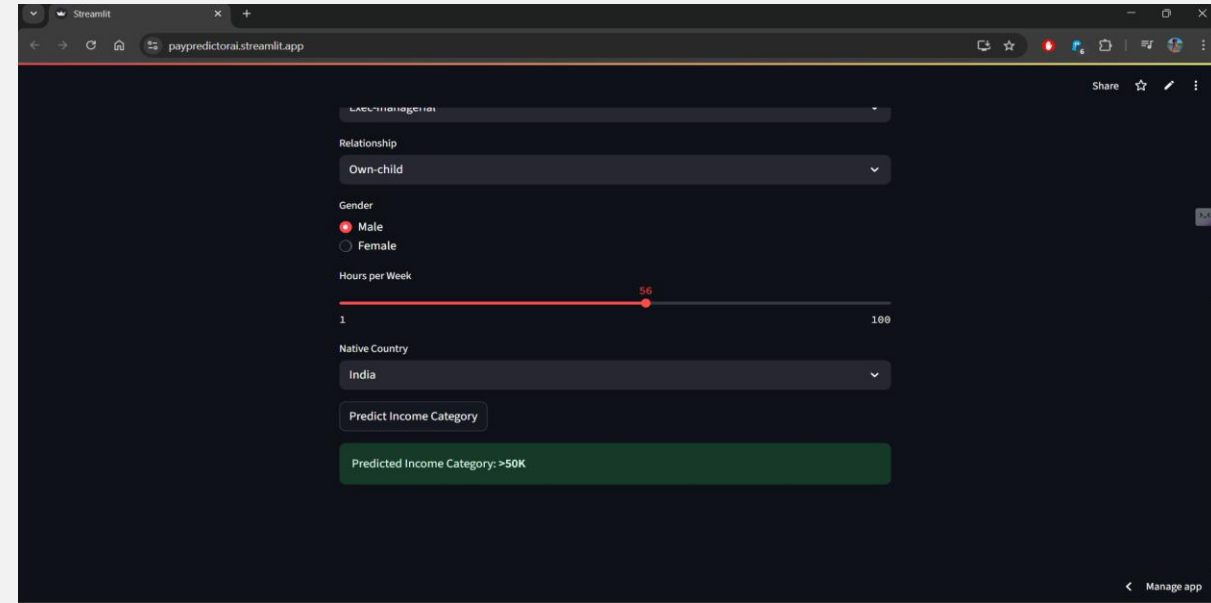
Figure: 6(a) – Snap of the application deployed using streamlit cloud

GitHub Repository Link: <https://github.com/Vidyadheesha-M-Pandurangi/PayPredictor-Emplyee-Salary-Prediction.git>

RESULT



The screenshot shows the 'PayPredictor' application interface. It has a dark theme with a sidebar on the left containing a 'Manage app' button. The main area has a title 'PayPredictor' with a brief instruction: 'Enter the following details to predict whether the income is >50K or <=50K'. Below this are several input fields: 'Age' (a slider set to 51), 'Workclass' (a dropdown menu set to 'Local-gov'), 'Education' (a dropdown menu set to 'Masters'), 'Marital Status' (a dropdown menu set to 'Never-married'), 'Occupation' (a dropdown menu set to 'Exec-managerial'), and 'Relationship' (a dropdown menu set to 'Own-child').



This screenshot shows the same application interface but with the 'Predict Income Category' button clicked. The button is now disabled. Below it, a green box displays the result: 'Predicted Income Category: >50K'. The input fields remain the same as in the previous screenshot.

Figure: 6(b) – Snap of the application deployed using streamlit cloud

Website URL: <https://paypredictorai.streamlit.app/>

CONCLUSION

This project, **PayPredictor**, was designed to develop a robust and interactive machine learning model that accurately predicts an employee's salary category ($\leq 50K$ or $> 50K$) based on various socio-economic and job-related features. By utilizing multiple machine learning algorithms such as **CatBoostClassifier**, **LightGBMClassifier**, **Quadratic Discriminant Analysis (QDA)**, and **Extra Trees Classifier**, we were able to compare and evaluate model performance through key metrics like accuracy and classification report. The best-performing model was then integrated into a **Streamlit web application** to provide a real-time, user-friendly salary classification interface.

The model's accuracy and generalization were greatly enhanced through effective **data preprocessing techniques** such as outlier removal, encoding of categorical features, and feature scaling. The use of **Pipeline and ColumnTransformer** allowed seamless integration of preprocessing and modeling, ensuring clean and reproducible workflows.

Findings

- Among all models tested, **CatBoostClassifier** yielded the highest accuracy, demonstrating its ability to handle categorical data efficiently and perform well on tabular datasets.
- The developed application provided consistent and interpretable results, making it valuable for HR analysts, career advisors, and data enthusiasts.

CONCLUSION

Challenges Encountered

- Managing missing or ambiguous data such as "?" entries in fields like occupation or native-country required careful preprocessing to prevent model bias.
- Handling outliers without losing meaningful data was crucial and involved visual analysis through boxplots.
- Ensuring compatibility between the trained model and the deployed application involved matching the input schema precisely, which initially led to ValueError due to missing columns.

The project successfully demonstrates how machine learning can be effectively used in salary prediction based on categorical and numerical inputs. The **PayPredictor** app serves as a practical and deployable solution for real-time salary classification, with scope for further innovation and feature enrichment.

FUTURE SCOPE

- Expanding the dataset with more real-world entries could further enhance model accuracy and generalization.
- Implementing cross-validation and hyperparameter tuning using GridSearchCV or Optuna can fine-tune the model's performance.
- Enhancing the UI of the Streamlit application with charts or explanation tools (like SHAP or LIME) could provide insights into why a particular prediction was made.
- Deploying on a more scalable platform (like AWS, GCP, or Azure) for production use and multi-user accessibility.
- Connecting PayPredictor with live HR management systems (like SAP SuccessFactors, Oracle HCM, or Workday) could automate and streamline predictions for recruitment and payroll processes.
- Extend the system to offer salary benchmarking—comparing employee compensation with industry averages or company standards—to assist in fair pay assessments.
- By adding NLP pipelines, resumes or job descriptions can be parsed to extract key features like skills, roles, and experience, which can then feed into the model for a more holistic salary prediction.
- Incorporate location-based adjustments such as city, state, or country, to reflect variations in salaries due to economic differences or living costs.

REFERENCES

- **Original source for the dataset:** <https://archive.ics.uci.edu/ml/datasets/adult>
- **Matplotlib:** <https://matplotlib.org/stable/contents.html>
- **Pandas:** <https://pandas.pydata.org/docs>
- **Numpy:** <https://numpy.org/doc>
- **scikit-learn:** <https://scikit-learn.org/stable/documentation.html>
- **CatBoost:** <https://catboost.ai/en/docs>
- **LightGBM:** <https://lightgbm.readthedocs.io/en/latest>
- **QuadraticDiscriminantAnalysis (QDA):** https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html
- **ExtraTreesClassifier:** <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>
- **Joblib:** <https://joblib.readthedocs.io/en/latest>
- **Streamlit:** <https://docs.streamlit.io>



THANK YOU