Annual income prediction model

Project 4
Sonia Morales
Elizabeth Romero
Isaac Silva
Santiago Morales

This project **transforms historical data** into actionable insights, showcasing the value of predictive modeling in **addressing societal challenges.**

Why This Project Matters

Real-World Relevance

- Understanding income patterns helps policymakers, businesses, and researchers analyze socio-economic trends.
- Offers insights into factors influencing income disparity in the U.S.

Practical Applications

- Targeted Marketing and Customer Segmentation
- Credit Risk Assessment
- Insurance Premium Calculation
- Personalized Financial Services
- E-Commerce and Subscription-Based Services
- Workforce Planning and Talent Acquisition
- Public Policy and Economic Planning

Bridging Technology and Society

- Highlights the power of data in making informed decisions.
- Promotes ethical use of predictive models to drive equity and opportunity.

Introduction

The US Adult Census dataset is a repository of 48,842 entries extracted from the 1994 US Census database.

In the first section, we focus on **data cleaning and optimization** to ensure its readiness for effective model training.

In the second section, we leverage this refined data to **develop predictive models** that determine whether an individual earned more or less than $50,000 in 1994. We then evaluate and compare these models to identify the approach that delivers the highest accuracy.

As a bonus, we implemented a user-friendly **form app** powered by our best-performing model, enabling real-time income predictions.

Basic data information

Each entry (row) contains the following information about an individual:

age: Integer > 0, representing an individual's age.

workclass: Employment status (e.g., Private, Self-employed, Government, etc.).

fnlwgt: Final weight, representing the estimated number of people the census entry corresponds to.

Education: Highest education level (e.g., Bachelors, High School, Doctorate, etc.).

Education num: Numeric representation of the highest education level (Integer > 0).

Marital status: Marital status (e.g., Married, Divorced, Never-married, etc.)

Occupation: General job type (e.g., Tech support, Sales, Armed Forces, etc.).

relationship: Role in household (e.g., Wife, Husband)

race: Person's race (e.g., White, Black, Indian)

sex: Biological sex (Male, Female).

capital-gain: Money earned from investments (≥0)

capital-loss: Money lost from investments (≥0)

hours-per-week: Weekly work hours

native-country: Birth country (e.g., US, Mexico)

the label(income): Earns <=50k or >50k USD annually

Data cleaning

Handling nulls and “?”
The dataset contained missing values and “?”
Fill in missing values with (Mode)

```
import numpy as np
# Replace "?" with NaN in the specified columns using .loc
X.loc[X['workclass'] == "?", 'workclass'] = np.nan
X.loc[X['occupation'] == "?", 'occupation'] = np.nan
X.loc[X['native-country'] == "?", 'native-country'] = np.nan

# Impute missing values with the most frequent value (mode) using .loc[]
X.loc[:, 'workclass'] = X['workclass'].fillna(X['workclass'].mode()[0])
X.loc[:, 'occupation'] = X['occupation'].fillna(X['occupation'].mode()[0])
X.loc[:, 'native-country'] = X['native-country'].fillna(X['native-country'].mode()[0])
```

Ensuring correct data types
The data types in the feature columns were verified to be the correct ones

```
X.loc[:, 'age'] = X['age'].astype(int)
X.loc[:, 'fnlwgt'] = X['fnlwgt'].astype(int)
X.loc[:, 'capital-gain'] = X['capital-gain'].astype(int)
X.loc[:, 'capital-loss'] = X['capital-loss'].astype(int)
X.loc[:, 'hours-per-week'] = X['hours-per-week'].astype(int)
```

Data cleaning

```python
# Perform one-hot encoding for categorical features
X = pd.get_dummies(X, columns=['workclass', 'education', 'marital-status', 'occupation',
                                'relationship', 'race', 'sex', 'native-country'], drop_first=True)
```

One-hot encoding
It was performed on categorical features
By doing this **education-num column** was dropped

```python
# Drop `education-num` since it seems redundant
X.drop('education-num', axis=1, inplace=True)
```

Duplicates in target columns
In the income columns 2 duplicates were found and replaced by the correct values.

```python
y = y.replace({"<=50K.": "<=50K", ">50K.": ">50K"})
```

Data model optimization

Feature Engineering

- Merging two columns variables
To optimize performance

-Drop Education-nums since it seems redundant

```
# Create a new feature 'net-capital-gain'
X['net-capital-gain'] = X['capital-gain'] - X['capital-loss']

# Drop the original 'capital-gain' and 'capital-loss' columns if no longer needed
X.drop(['capital-gain', 'capital-loss'], axis=1, inplace=True)
```

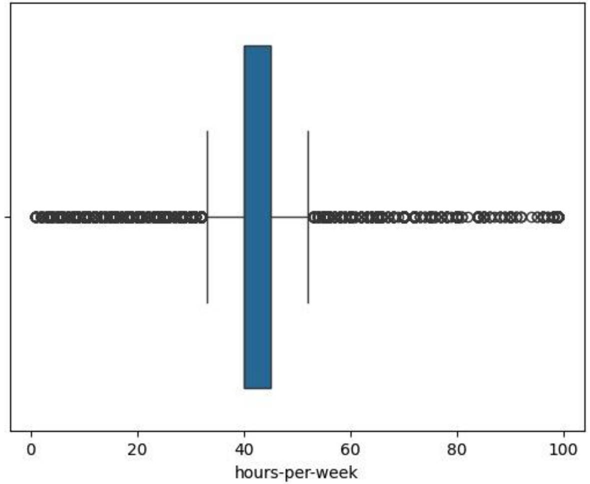
```
# Drop `education-num` since it seems redundant
X.drop('education-num', axis=1, inplace=True)
```

Handling Outliers

```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot to check for outliers in 'hours-per-week'
sns.boxplot(x=X['hours-per-week'])
# Remove outliers in 'hours-per-week'
X = X[X['hours-per-week'] <= 80]

plt.show()
```



K means Clustering

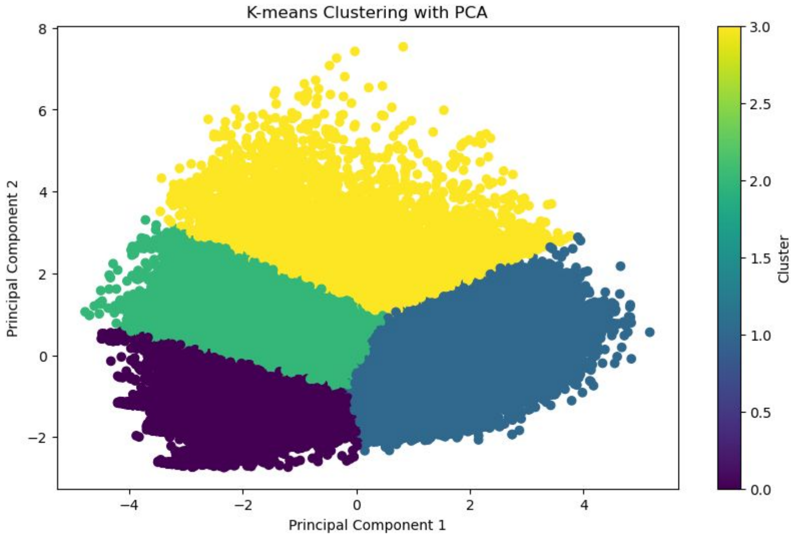
Using KMeans with PCA enables efficient, interpretable, and scalable clustering for this high-dimensional dataset. It focuses on relevant patterns, reduces noise, and ensures clusters are well-separated, providing actionable insights with reduced complexity.

Cluster 0 - Purple :

- **Youngest and least experienced** group.
- Overwhelmingly private sector (~96%).
- Lowest net capital gains and income (~0.85% earn >50k).
- **Represents entry-level workers** with limited wealth accumulation.

Cluster 1 - Blue:

- **Oldest and wealthiest** group.
- Highest net capital gains (~1812) and income (~46.3% earn >50k).
- Longest working hours
- Mix of private and self-employed workers.
- **Represents high earners, experienced professionals** with diverse job types.



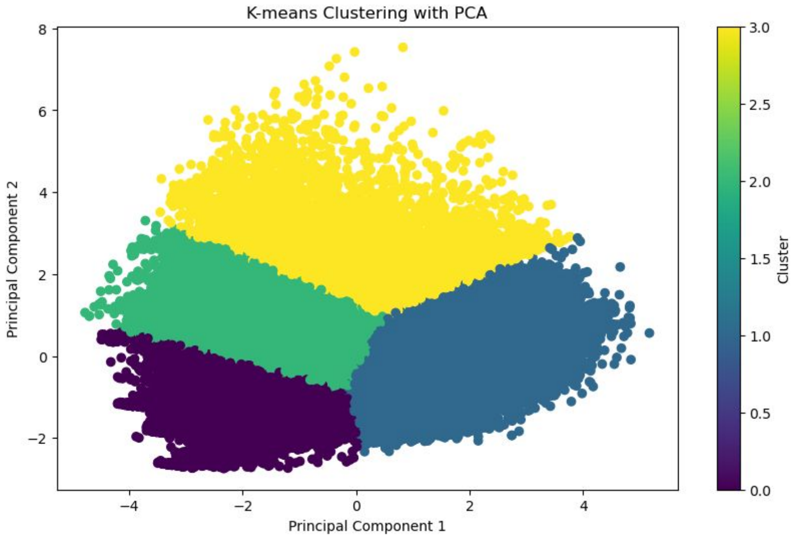
K means Clustering

Cluster 2 - Green:

- **Middle-aged group** in predominantly private employment (~86%).
- Low income (~3.7% earn >50k) and moderate work hours (~38.2).
- Likely represents **middle-income earners** with lower financial gains.

Cluster 3 - Yellow:

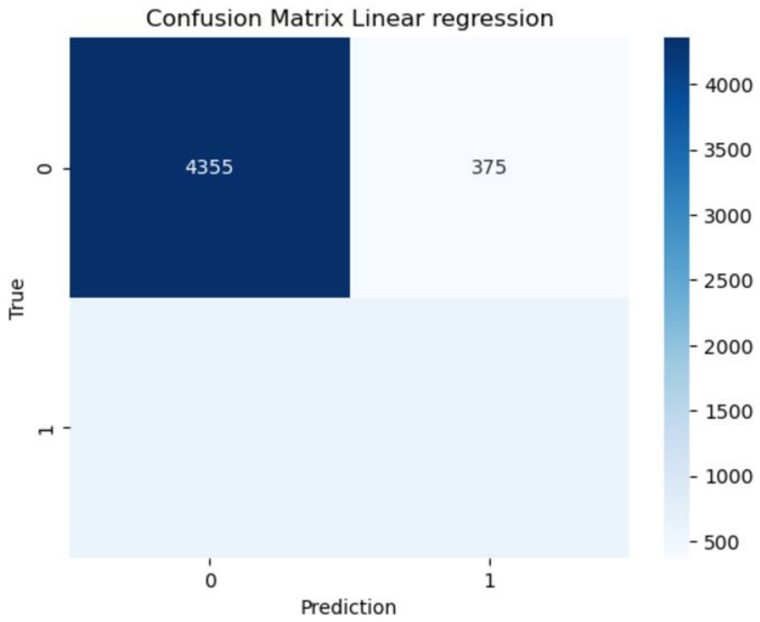
- **Older and diverse employment**.
- High representation in **Local/State government** jobs.
- Moderate capital gains (~876) and income (~19.9% earn >50k).
- Likely represents **government employees** with stable, moderate earnings.



Logistic regression

Classification Report:

	precision	recall	f1-score	support
<=50K	0.88	0.92	0.90	4730
>50K	0.62	0.51	0.56	1202
accuracy			0.84	5932
macro avg	0.75	0.71	0.73	5932
weighted avg	0.83	0.84	0.83	5932

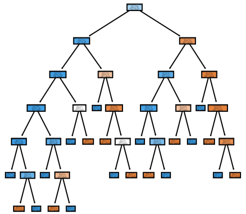



```
Accuracy: 0.8465
Classification Report:
              precision    recall  f1-score   support

       <=50K       0.88      0.94      0.91      6649
        >50K       0.62      0.45      0.52      1512

    accuracy                           0.85      8161
   macro avg       0.75      0.69      0.71      8161
weighted avg       0.83      0.85      0.84      8161

Confusion Matrix:
[[6226  423]
 [ 830  682]]
```



Accuracy: 84.65%

Key Insights:

- The model is great at identifying people earning <=50K, with a recall of **94%** (it correctly identified most cases).
- However, it struggles with >50K predictions, achieving only **45% recall**, meaning it missed quite a few higher earners.
- The overall weighted scores indicate that the model balances predictions well but favors the majority class (<=50K).

Confusion Matrix:

- **6226** people earning <=50K were correctly classified, but **423** were misclassified as >50K.
- For >50K, **830** were misclassified as <=50K, while only **682** were correctly identified.

Neural Network

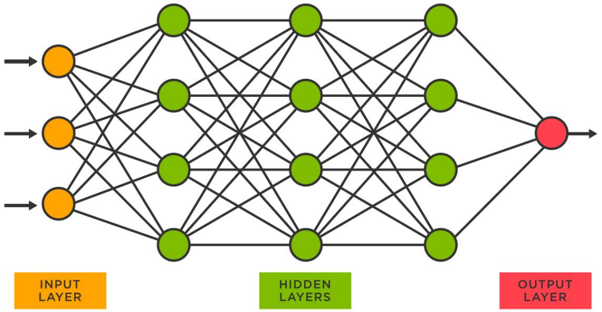
Accuracy: **0.8450**

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90      6649
           1       0.57      0.56      0.57      1512

    accuracy                           0.84      8161
   macro avg       0.74      0.73      0.73      8161
weighted avg       0.84      0.84      0.84      8161


Confusion Matrix:
[[6015  634]
 [ 665  847]]
```



Accuracy: 84.5%

Key Insights:

- Similar to Random Forest, the model performs very well for <=50K cases, with **90% precision and recall**.
- For >50K predictions, the precision and recall drop to about **57%**, meaning it has trouble differentiating higher earners.
- The accuracy is slightly lower than Random Forest, but it maintains a good balance in predictions.

Confusion Matrix:

- **6015** <=50K cases were correctly classified, but **634** were wrongly predicted as >50K.
- For >50K, **847** were correctly classified, but **665** were missed.

Keras

XGBoost

```
Accuracy: 0.8585

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      6649
           1       0.67      0.46      0.55      1512

    accuracy                           0.86      8161
   macro avg       0.78      0.71      0.73      8161
weighted avg       0.85      0.86      0.85      8161


Confusion Matrix:
[[6308  341]
 [ 814  698]]
```

Accuracy: 85.85%
Key Insights:

- XGBoost outperformed the other models overall, with the highest accuracy.
- It has strong precision for >50K predictions (**67%**) but still struggles with recall for this group (**46%**), meaning it misses some higher earners.
- It balances predictions better than the other models, achieving the best macro-average F1 score.

Confusion Matrix:

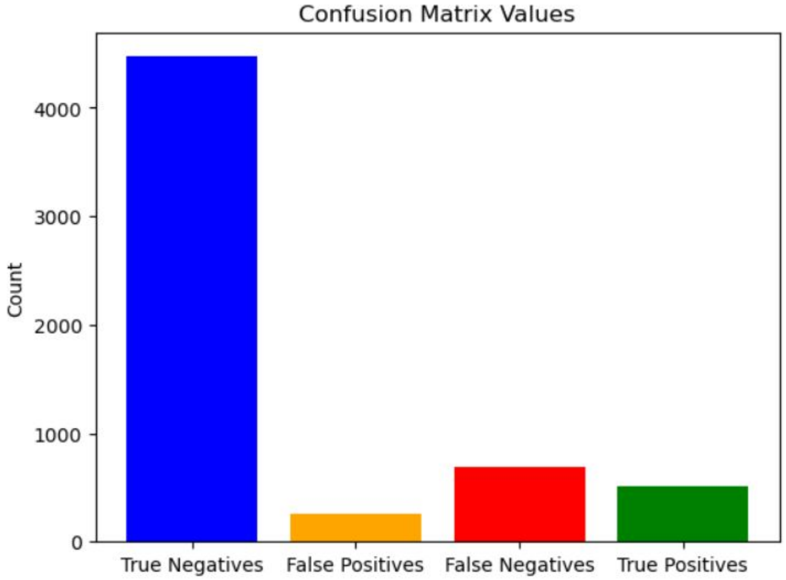
- **6308** <=50K cases were correctly classified, with only **341** misclassified as >50K.
- For >50K, **698** were correctly predicted, while **814** were missed.

Support Vector Classification

Support Vector Classification uses **support vectors** to predict the target values of income

Classification Report

	precision	recall	f1-score	support
<=50K	0.87	0.95	0.90	4730
>50K	0.66	0.43	0.52	1202
accuracy			0.84	5932
macro avg	0.77	0.69	0.71	5932
weighted avg	0.83	0.84	0.83	5932



Why our models are predicting <50K better?

1. Class Imbalance

In the dataset, there are significantly more people earning <=50K than >50K:

- **<=50K**: 6649 samples
- **>50K**: 1512 samples

This imbalance means the models see a lot more examples of the <=50K group during training, making them better at recognizing patterns for this majority class. However, since there are fewer examples of >50K, the models don't learn as effectively to identify these cases, leading to lower recall and precision for >50K predictions.

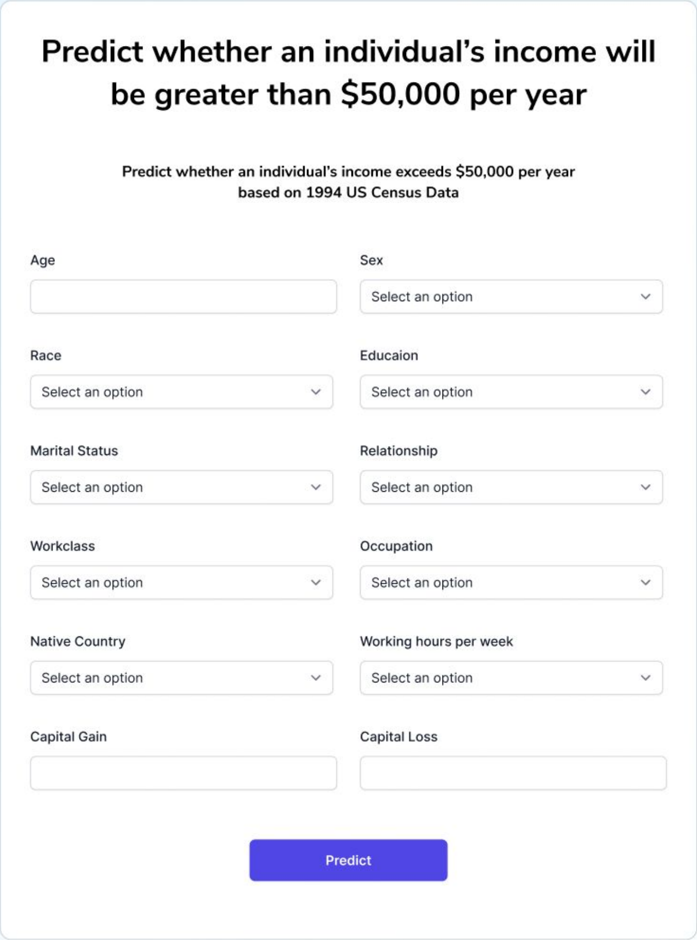
2. Decision Bias Toward the Majority Class

Most machine learning models aim to maximize overall accuracy. To do this, they tend to focus on the majority class (<=50K) because misclassifying fewer higher earners (>50K) won't impact the accuracy as much. For example:

- If the model guesses <=50K for everyone, it would still achieve ~81% accuracy because most people in the dataset earn <=50K.

This bias makes the models less likely to predict >50K, even when the data might indicate it.

Flask App


Predict whether an individual's income will be greater than $50,000 per year
Predict whether an individual's income exceeds $50,000 per year based on 1994 US Census Data
Age
Sex
Select an option
Race
Select an option
Educaion
Select an option
Marital Status
Select an option
Relationship
Select an option
Workclass
Select an option
Occupation
Select an option
Native Country
Select an option
Working hours per week
Select an option
Capital Gain
Capital Loss
Predict

Thank you!

Requirements

Data Model Implementation (25 points)

- A Python script initializes, trains, and evaluates a model (10 points)
- The data is cleaned, normalized, and standardized prior to modeling (5 points)
- The model utilizes data retrieved from SQL or Spark (5 points)
- The model demonstrates meaningful predictive power at least 75% classification accuracy or 0.80 R-squared. (5 points)

Data Model Optimization (25 points)

- The model optimization and evaluation process showing iterative changes made to the model and the resulting changes in model performance is documented in either a CSV/Excel table or in the Python script itself (15 points)
- Overall model performance is printed or displayed at the end of the script (10 points)

GitHub Documentation (25 points)

- GitHub repository is free of unnecessary files and folders and has an appropriate .gitignore in use (10 points)
- The README is customized as a polished presentation of the content of the project (15 points)

Group Presentation (25 points)

- All group members speak during the presentation. (5 points)
- Content, transitions, and conclusions flow smoothly within any time restrictions. (5 points)
- The content is relevant to the project. (10 points)
- The presentation maintains audience interest. (5 points)