

# income-predictor-analysis

November 22, 2025

## 0.1 Predicting income category from socioeconomic characteristics

by Luke Ni, Michael Oyatsi, Nishanth Kumarasamy & Shruti Sasi

```
[1]: #imports
import numpy as np
import pandas as pd
import altair as alt
import altair_ally as aly
import shap
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.preprocessing import FunctionTransformer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report
from sklearn.metrics import make_scorer, precision_score, recall_score, f1_score, accuracy_score
from sklearn.model_selection import cross_validate

# Simplify working with large dataset in altair_ally
aly.alt.data_transformers.enable('vegafusion')
```

```
/Users/michaeloyatsi/miniforge3/envs/DSCI_522_project_env/lib/python3.12/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
from .autonotebook import tqdm as notebook_tqdm
```

```
[1]: DataTransformerRegistry.enable('vegafusion')
```

## 0.2 Executive Summary

We built a classification model to predict an individual's income group, split by whether they are high earners ( $>$  USD 50,000) or low earners ( $<$  USD 50,000). Using a Logistic Regression classifier, our model accuracy was 80% on unseen test data with an associated F1 score of 0.663. To address the class imbalance in the data, we used a balanced weight approach while building our model. We also sought to understand what socioeconomic characteristics play a the biggest role in determining an individuals income group. Using SHAP analysis, our findings show that of the features in our model, Marital Status, Age & Education are the biggest drivers of a High Income output.

While the Logistic Regression classifier was chose to easier identify the socioeconomic features that are drivers of high income, we see an opportunity to use an ensemble model such as Random Forest Classification to improve the model's prediction metrics.

## 0.3 Introduction

How is an individual's income affected by other socioeconomic factors? This is the question our team set out to investigate. Socioeconomic status here is defined as a way of describing people based on their education, income and type of job (National Cancer Institute, n.d.). With the diversity of backgrounds that can exist in society, we set out to understand what factors contribute most to an individuals income.

In this analysis, we use machine learning to predict whether an individuals income is above or below \$50,000. As the government sets out massive investment in Canadian societies to improve the lives of citizens(Housing, Infrastructure and Communities Canada, 2025), we envision our analysis as a means of providing insights to the government as to what investments can drive the best chances of improving an individuals life. The persistent income and wealth inequality increase presents a strong case for prudent investing to improve lives across all Canadians. (Yassin, Petit, & Abraham, 2024)

## 0.4 Methods

#### Data

For our dataset, we use the Adult dataset sourced from the UC Irvine Machine Learning Repository (Becker & Kohavi, 1996). The dataset contains 14 features obtained from census data to describe an individuals attributes. The target is a categorical column comprised of a binary outcome of whether an individual earns more than USD 50,000( $>50K$ ) or USD 50,000 or less ( $\leq 50K$ ). The data and the descriptions fo the corresponding attributes can be explored using this [link](#)

## 0.5 Exploratory Data Analysis

Prior to model fitting and feature selection, we first perform EDA to understand the distribution of our features as it relates to our target.

The code chunk below imports our dataset.

```
[2]: # Import the data from the UCI Repository.
```

```
from ucimlrepo import fetch_ucirepo
```

```

# fetch dataset
adult = fetch_ucirepo(id=2)

# data (as a pandas dataframe)
adult_df = pd.concat([adult.data.features, adult.data.targets], axis=1)

# Rename target values
adult_df.income = adult_df.income.replace(to_replace=['<=50K.', '>50K.'],
↪value=['<=50K', '>50K'])

# Combine all married groups in marital status to one group.
adult_df['marital-status'] = adult_df['marital-status'].
↪replace(to_replace=r'^Married\b.*', value='Married', regex=True)

# Display First observations of the dataset
adult_df.head(5)

```

```

[2]:   age      workclass  fnlwgt  education  education-num  marital-status  \
0   39      State-gov   77516   Bachelors              13  Never-married
1   50  Self-emp-not-inc   83311   Bachelors              13      Married
2   38      Private  215646   HS-grad               9      Divorced
3   53      Private  234721     11th                7      Married
4   28      Private  338409   Bachelors              13      Married

      occupation  relationship   race   sex  capital-gain  \
0  Adm-clerical  Not-in-family  White  Male         2174
1  Exec-managerial      Husband  White  Male           0
2  Handlers-cleaners  Not-in-family  White  Male           0
3  Handlers-cleaners      Husband  Black  Male           0
4  Prof-specialty      Wife  Black  Female           0

  capital-loss  hours-per-week  native-country  income
0           0             40  United-States  <=50K
1           0             13  United-States  <=50K
2           0             40  United-States  <=50K
3           0             40  United-States  <=50K
4           0             40         Cuba  <=50K

```

**Train-Test-Split: Obey the Golden Rule** Before proceeding with further EDA and visualization of the data, we split and stash a test set from our data in order to evaluate our model performance on unseen data in accordance with the principles of the Golden rule of machine learning.

```

[3]: # Create a Test train split of the data
adult_train, adult_test = train_test_split(adult_df, test_size=0.3,
↪random_state=522)

```

**Discern Features & Strategize Missing Data** With the split, complete we review on the `adult_train` data to understand the statistics of the numerical features and to investigate on the presence of null values.

```
[4]: # Investigate quality of the data
adult_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 34189 entries, 21696 to 3360
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   age                   34189 non-null  int64
 1   workclass              33508 non-null  object
 2   fnlwgt                 34189 non-null  int64
 3   education              34189 non-null  object
 4   education-num          34189 non-null  int64
 5   marital-status         34189 non-null  object
 6   occupation              33506 non-null  object
 7   relationship           34189 non-null  object
 8   race                   34189 non-null  object
 9   sex                    34189 non-null  object
10   capital-gain           34189 non-null  int64
11   capital-loss           34189 non-null  int64
12   hours-per-week         34189 non-null  int64
13   native-country         34016 non-null  object
14   income                 34189 non-null  object
dtypes: int64(6), object(9)
memory usage: 4.2+ MB
```

```
[5]: # Summarize null values
adult_train.isna().sum()
```

```
[5]: age                0
workclass              681
fnlwgt                 0
education              0
education-num          0
marital-status         0
occupation             683
relationship           0
race                   0
sex                    0
capital-gain           0
capital-loss           0
hours-per-week         0
native-country         173
income                 0
```

dtype: int64

Three columns in the dataset contain NaN values. Further review of the dataset also shows that the same columns have values that are missing, but populated by a question mark to indicate lack of information. Given the quantity of observations with None or Missing values, we create a separate value **Unknown** in order to avoid adding our own influence to the data using imputation such as Simple Imputation with most frequent values in the feature.

```
[6]: # Visualize distribution of features containing incomplete information values
obj_df = adult_train.select_dtypes(include='object')
for col in obj_df.columns:
    if obj_df[col].str.contains('?', regex=False).any():
        print (col)
```

workclass  
occupation  
native-country

```
[7]: # Visualize distributions of categorical variables in the features with missing
↳ data.
for col in obj_df.columns:
    if obj_df[col].str.contains('?', regex=False).any():
        print (obj_df[col].value_counts())
```

workclass

Private	23697
Self-emp-not-inc	2705
Local-gov	2193
State-gov	1391
?	1323
Self-emp-inc	1190
Federal-gov	987
Without-pay	15
Never-worked	7

Name: count, dtype: int64

occupation

Prof-specialty	4340
Craft-repair	4265
Exec-managerial	4265
Adm-clerical	3923
Sales	3895
Other-service	3432
Machine-op-inspct	2094
Transport-moving	1648
Handlers-cleaners	1404
?	1328
Tech-support	1033
Farming-fishing	1023
Protective-serv	680

Priv-house-serv	167
Armed-Forces	9
Name: count, dtype: int64	
native-country	
United-States	30725
Mexico	680
?	400
Philippines	200
Germany	141
Puerto-Rico	129
Canada	126
India	110
El-Salvador	107
Cuba	97
England	83
South	81
Italy	78
China	78
Dominican-Republic	74
Jamaica	72
Japan	70
Poland	64
Guatemala	61
Vietnam	54
Columbia	54
Haiti	51
Portugal	47
Iran	43
Taiwan	39
Nicaragua	37
Ecuador	36
Peru	35
Greece	32
France	26
Ireland	20
Hong	19
Cambodia	19
Yugoslavia	18
Laos	18
Thailand	18
Outlying-US(Guam-USVI-etc)	17
Scotland	16
Hungary	14
Trinidad&Tobago	14
Honduras	12
Holand-Netherlands	1
Name: count, dtype: int64	

```
[8]: # Reinforce conversion of columns to str object.
# for col in adult_train.select_dtypes(include='object'):
#     adult_train[col] = adult_train[col].astype(str)

# Replace missing data with NA
#adult_train = adult_train.apply(lambda col: col.str.strip() if col.
#    dtype=='object' else col)
adult_train = adult_train.replace('?', np.nan)

# Perform Simple Imputation
simple_imp = SimpleImputer(missing_values = np.nan, strategy='most_frequent')
adult_train_imp = pd.DataFrame(simple_imp.fit_transform(adult_train),
                               index=adult_train.index,
                               columns=adult_train.columns)

# Recast numerical features to int data types after Impute
adult_train_imp = adult_train_imp.astype({'age': 'int64',
                                           'fnlwgt': 'int64',
                                           'capital-gain': 'int64',
                                           'capital-loss': 'int64',
                                           'hours-per-week': 'int64'})

# Confirm all missing values have been imputed
adult_train_imp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 34189 entries, 21696 to 3360
Data columns (total 15 columns):
#   Column                Non-Null Count  Dtype
---  -
0   age                   34189 non-null  int64
1   workclass             34189 non-null  object
2   fnlwgt                34189 non-null  int64
3   education             34189 non-null  object
4   education-num         34189 non-null  object
5   marital-status        34189 non-null  object
6   occupation            34189 non-null  object
7   relationship          34189 non-null  object
8   race                  34189 non-null  object
9   sex                   34189 non-null  object
10  capital-gain           34189 non-null  int64
11  capital-loss           34189 non-null  int64
12  hours-per-week         34189 non-null  int64
13  native-country         34189 non-null  object
14  income                 34189 non-null  object
dtypes: int64(5), object(10)
```

memory usage: 4.2+ MB

**Univariate Distribution of The Quantitative Variables** *Note - Visualization of the distributions below using the `altair-ally` package is performed with code adapted from UBC's DSCI-573: Feature and Model Selection Course. Reference on the Altair Ally package can be found in using [this external link](#)*

We first investigate the distribution of the dataset's quantitative variables age against their income bracket. From the plots below, we pay special focus on the age distribution of the respondents. Both distributions are right-skewed. Income earners at or below USD 50,000 skew younger than fellow respondents earning above USD 50,000.

Of note also is the age distribution of hours worked per week, with most respondents in both income brackets reporting about 40 hours per week. The `fnlweight` feature is a final numerical value representing the final weight of the record. This value can be viewed as the number of people represented by the row. Without further breakdown on the methods or derivation of this value, we chose to ignore it in our analysis.

Similarly, no in depth data is provided on the `capital-loss` and `capital-gain` features, but these fields may have strong predictive value for identifying higher-income earners, so we consider creating a binary indicator (like `has_capital_gain`) to capture the signals.

```
[9]: aly.dist(adult_train_imp, color='income')
```

```
[9]: alt.ConcatChart(...)
```

**Univariate Distribution of the Categorical Variables** We also review the distribution of select categorical variables below.

From the first histogram of `income` distribution, we can see that the dataset contains more records of low income earners compared to high income earners, a ration of about 3:1.

Reviewing the marital status of distribution, we can see that the distribution of high income earners is concentrated primarily on married respondents with scant representation the other marital status groups. Note that the original dataset contains 3 distinc married groups: `Married-AF-spouse` for respondents whose partners are in the Army, `Married-civ-spouse` for individuals married to civilian spouses & `Married-spouse-absent`. For simplification, all values have been combined into one variable `Married`.

When analysing the `workclass` feature, a feature to classify the respondents' employer, we notice that high income earners are represented primarily in the private sector and less so in other employer categories.

While the distribution of the `occupation` feature is less conclusive, we see that high income earners in `exec-managerial` and `prof-specialty`, executive management and professional specialties respectively. Likewise when reviewing the `education` feature, we can see that high income earners tend to have at least some college education and are barely present in respondents who did not finish high school (For clarity, 12th grade is the last year of high school)

In order to prevent propagating inherent societal biases in our model, the following categories are not considered for feature or model selection: `race`.

Moreover, the `relationship` feature, which represents the relationship the observation has relative to others is not considered as the useful information is encoded within the respondent's marital status.

We also exclude the `native-country` from our visualization and feature. The overwhelming majority of the respondents are American-born and with little information on other information regarding the foreign-born respondents (e.g. how long they have been in the USA), we exclude this feature from our model.

```
[10]: # Look at the univariate distributions (counts) for the categorical variables

# Changing churn to an object dtype just for the data passed to the chart
aly.dist(adult_train_imp.select_dtypes(include='object').
    ↪drop(columns=['relationship', 'sex',
    ↪'education-num', 'race',
    ↪'native-country']),
    dtype='object', color='income')
```

```
[10]: alt.ConcatChart(...)
```

## 0.6 Features & Model Selection

### 0.6.1 Pre-processing pipeline

The adult dataset has various types of features: numeric, categorical, binary.

Feature	Type	Transformation
age	Integer	Scaling with StandardScaler
workclass	Categorical	imputation, one-hot encoding
fnlwgt	Integer	drop
education	Categorical	drop
education-num	Integer	no transformation needed
marital-status	Categorical	one-hot encoding
occupation	Categorical	imputation, one-hot encoding
relationship	Categorical	drop
race	Categorical	drop
sex	Binary	one-hot encoding with drop= <code>if_binary</code>
capital-gain	Integer	drop
capital-loss	Integer	drop
hours-per-week	Integer	Scaling with StandardScaler
native-country	Categorical	imputation, one-hot encoding

```
[11]: def binary_flag(x):
    # Binary conversion for captital features
    return (x > 0).astype(int)

# Features
numeric_features = ["age", "hours-per-week", "education-num"]
categorical_features = ["workclass", "marital-status", "occupation",
    ↪"native-country"]
binary_features = ["sex"]
drop_features = ["fnlwgt", "education", "relationship", "race", "capital-gain",
    ↪"capital-loss"]
capital_features = ["capital-gain", "capital-loss"]
target = "income"
```

```

# Transformers
numeric_transformer = StandardScaler()
capital_transformer = FunctionTransformer(binary_flag,
    ↳feature_names_out="one-to-one")
binary_transformer = OneHotEncoder(drop="if_binary", dtype=int)
categorical_transformer = make_pipeline(
    SimpleImputer(missing_values=np.nan, strategy='constant',
    ↳fill_value='Unknown'),
    # SimpleImputer(missing_values = np.nan, strategy='most_frequent'),
    OneHotEncoder(handle_unknown="ignore", sparse_output=False)
)

# Preprocessor
preprocessor = make_column_transformer(
    (numeric_transformer, numeric_features),
    (categorical_transformer, categorical_features),
    (binary_transformer, binary_features),
    (capital_transformer, capital_features),
    ("drop", drop_features)
)
preprocessor

```

```

[11]: ColumnTransformer(transformers=[('standardscaler', StandardScaler(),
    ['age', 'hours-per-week', 'education-num']),
    ('pipeline',
    Pipeline(steps=[('simpleimputer',
    SimpleImputer(fill_value='Unknown',
    strategy='constant')),
    ('onehotencoder',
    OneHotEncoder(handle_unknown='ignore',
    sparse_output=False))]),
    ['workclass', 'marital-status', 'occupation',
    'native-country']),
    ('onehotencoder',
    OneHotEncoder(drop='if_binary',
    dtype=<class 'int'>),
    ['sex']),
    ('functiontransformer',
    FunctionTransformer(feature_names_out='one-to-
one',
    func=<function binary_flag
at 0x168239260>),
    ['capital-gain', 'capital-loss']),
    ('drop', 'drop',
    ['fnlwgt', 'education', 'relationship', 'race',
    'capital-gain', 'capital-loss'])])

```

```
[12]: # Replacing ? with NULL values in test dataset
adult_test = adult_test.replace('?', np.nan)

# splitting features and target
X_train = adult_train.drop(columns=target)
X_test = adult_test.drop(columns=target)
y_train = adult_train[target]
y_test = adult_test[target]
```

## 0.6.2 Fit a model

```
[13]: # creating a pipeline with preprocessing + LogisticRegression
model = make_pipeline(
    preprocessor,
    LogisticRegression(class_weight='balanced', max_iter=1000,
↳ random_state=522, C=10)
)

# fit model on the entire training set
model.fit(X_train, y_train)
```

```
[13]: Pipeline(steps=[('columntransformer',
                        ColumnTransformer(transformers=[('standardscaler',
                                                         StandardScaler(),
                                                         ['age', 'hours-per-week',
                                                         'education-num']),
                                                         ('pipeline',
                                                         Pipeline(steps=[('simpleimputer',
                                                         SimpleImputer(fill_value='Unknown',
                                                         strategy='constant')),
                                                         ('onehotencoder',
                                                         OneHotEncoder(handle_unknown='ignore',
                                                         sparse_output=False))]),
                                                         ['workclass',
                                                         'marital-s...
                                                         ['sex']],
                                                         ('functiontransformer',
                                                         FunctionTransformer(feature_names_out='one-to-one',
                                                         func=<function binary_flag at 0x168239260>),
                                                         ['capital-gain',
                                                         'capital-loss']),
                                                         ('drop', 'drop',
                                                         ['fnlwt', 'education',
                                                         'relationship', 'race',
                                                         'capital-gain',
                                                         'capital-loss'])])),
                        ('logisticregression',
```

```
LogisticRegression(C=10, class_weight='balanced',
                    max_iter=1000, random_state=522)))]
```

### 0.6.3 Model Evaluation

**Accuracy, Precision, Recall, F1-Score** Our Logistic Regression model demonstrates robust generalization with a test accuracy of **80.2%**, mirroring the cross-validation mean of **79.8%** and indicating no overfitting. By employing balanced class weights to address dataset imbalance, the model strategically prioritizes Recall (82.7%) over Precision (55.4%) for the high-income class (>50K). This configuration results in an F1-score of 0.66, confirming that while the model successfully identifies the vast majority of high earners, it accepts a higher rate of false positives to ensure potential high-income individuals are rarely missed.

```
[14]: # Logistic regression
classification_metrics = {
    "accuracy": "accuracy",
    "precision": make_scorer(precision_score, pos_label=">50K"),
    "recall": make_scorer(recall_score, pos_label=">50K"),
    "f1": make_scorer(f1_score, pos_label=">50K")
}

cross_val_results = {}
cross_val_results['model'] = pd.DataFrame(cross_validate(
    model,
    X_train,
    y_train,
    return_train_score=True,
    scoring=classification_metrics
)).agg(['mean', 'std']).round(3).T

# Show the train and validation scores
print("Test Score: ", model.score(X_test, y_test))
cross_val_results['model']
```

Test Score: 0.8021565549716781

```
[14]:
```

	mean	std
fit_time	0.201	0.014
score_time	0.072	0.007
test_accuracy	0.798	0.007
train_accuracy	0.800	0.002
test_precision	0.554	0.011
train_precision	0.556	0.002
test_recall	0.827	0.007
train_recall	0.832	0.002
test_f1	0.663	0.009
train_f1	0.667	0.002

**Confusion matrix & Classification report** The matrix confirms our high-recall strategy: the model successfully identifies the vast majority of high-income earners (minimizing False Negatives), though this aggressiveness results in more lower-income individuals being incorrectly flagged (higher False Positives).

```
[15]: # Confusion matrix for the logistic regression
confmat_logreg = ConfusionMatrixDisplay.from_estimator(
    model,
    X_test,
    y_test,
    normalize='all'
)

# Show the matrix
print(confmat_logreg)

# Classification Report

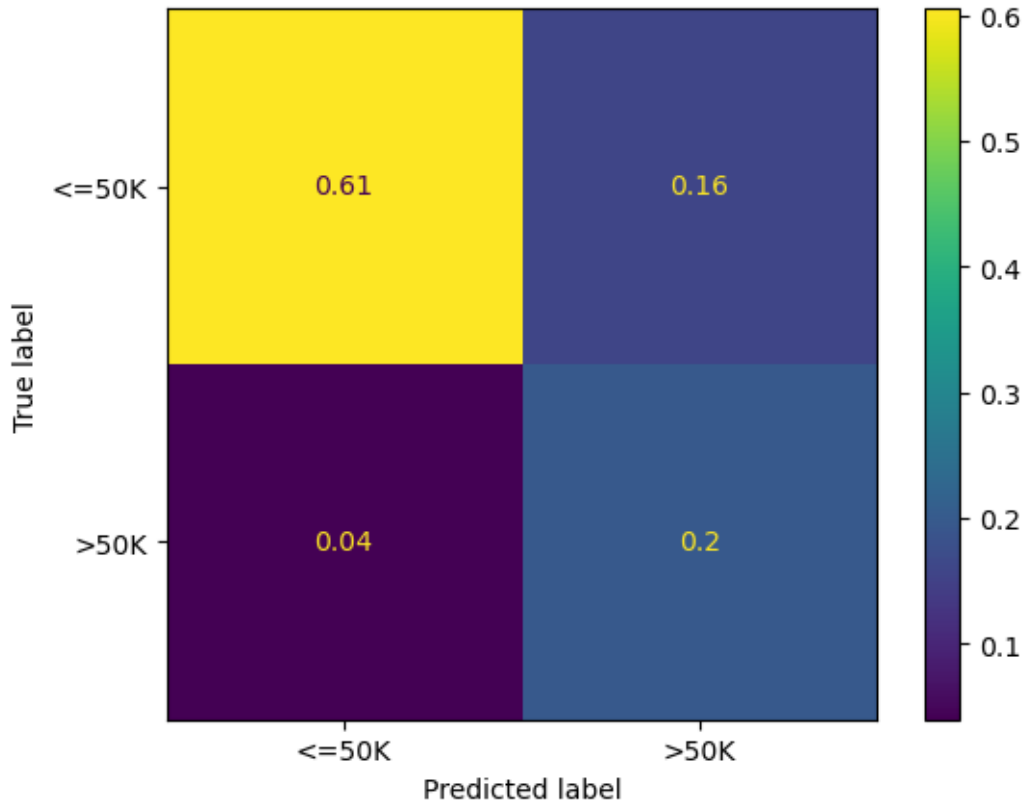
print("Classification Report: ")
pd.DataFrame(classification_report(
    y_test,
    model.predict(X_test),
    target_names=['<=50K', '>50K'],
    output_dict=True
)).transpose()
```

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay object at 0x16835fe60>

Classification Report:

```
[15]:
```

	precision	recall	f1-score	support
<=50K	0.938333	0.792974	0.859551	11187.000000
>50K	0.554530	0.831795	0.665436	3466.000000
accuracy	0.802157	0.802157	0.802157	0.802157
macro avg	0.746431	0.812384	0.762494	14653.000000
weighted avg	0.847549	0.802157	0.813636	14653.000000



#### 0.6.4 Model Explainability (using SHAP)

SHAP values interpret the model's decisions by assigning an 'importance score' to every feature for every prediction. This allows us to see exactly which factors pushed a specific individual's prediction towards the high-income or low-income category.

```
[16]: # A. Extract steps from the pipeline
log_reg_model = model.named_steps['logisticregression']
preprocessor_step = model.named_steps['columntransformer']

# B. Transform Data for SHAP
X_train_transformed = preprocessor_step.transform(X_train)
X_test_transformed = preprocessor_step.transform(X_test)

# C. Get Feature Names
feature_names = preprocessor_step.get_feature_names_out()

# D. Create Explainer
X_train_summary = shap.sample(X_train_transformed, 100)
explainer = shap.LinearExplainer(log_reg_model, X_train_summary)
```

```

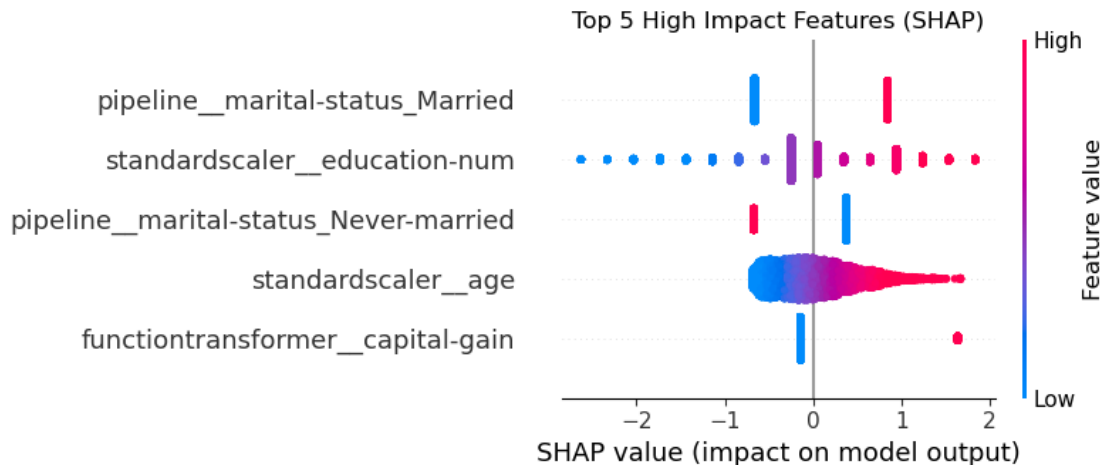
# E. Calculate SHAP Values
shap_values = explainer.shap_values(X_test_transformed)

# F. Handle Binary Classification Output - Class 1 (>50K)
if isinstance(shap_values, list):
    vals_to_plot = shap_values[1]
else:
    vals_to_plot = shap_values

# G. Plot Summary with only Top 5 Features
plt.figure(figsize=(10, 6))
plt.title("Top 5 High Impact Features (SHAP)")

shap.summary_plot(
    vals_to_plot,
    X_test_transformed,
    feature_names=feature_names,
    max_display=5 # Top 5 important features
)

```



### 0.6.5 Results:

#### 1. Model Performance:

- **Accuracy:** The model achieved an accuracy of 80% on the test set.
- **Class Imbalance Handling:** By using `class_weight='balanced'`, we prioritized identifying high-income earners (>50K). This likely resulted in higher Recall for the >50K class (catching more high earners) potentially at the cost of some Precision (more false positives).
- **Confusion Matrix Analysis:** As seen in the matrix, the model correctly identified high-income individuals, while missing some False positives.

**2. Feature Importance (Explainability)** Using SHAP analysis, we identified the key drivers of income:

- **Marital Status:** Being Married-civ-spouse is often the strongest positive predictor of high income (indicated by the long red bar pushing to the right in the SHAP plots).
- **Age:** The dependence plot shows a positive correlation between age and income up to a certain point (likely 50-60 years old), after which it may plateau or decrease.
- **Education:** Higher education-num consistently pushes predictions toward the >50K category.
- **Capital Gain:** Individuals with significant capital gains are overwhelmingly classified as high income.

### 0.6.6 Conclusion

This analysis successfully established a robust predictive model for income classification, leveraging logistic regression to identify the key drivers of economic disparities. By intentionally designing a pipeline that addresses class imbalance, our model demonstrates a high sensitivity to detecting high-income earners, ensuring that significant predictors of wealth are not overlooked. While we prioritized ethical fairness by excluding explicit racial and relationship identifiers, the model’s performance confirms that other structural factors—specifically education, marital status, and career stability—remain powerful proxies for economic success in the current landscape.

From a socioeconomic perspective, our results align closely with established economic theory. **Education** emerged as a dominant differentiator, validating the concept of human capital where higher investment in skills directly correlates with earning potential. Similarly, **Marital Status** proved to be a substantial predictor, likely reflecting the economic stability often associated with dual-income households or the “marriage premium” phenomenon observed in labor economics. **Age** also displayed a strong positive trend, illustrating the natural accumulation of experience and seniority over a career trajectory, though this effect naturally plateaus as individuals near retirement.

### 0.6.7 References

National Cancer Institute. (n.d.). Socioeconomic status. In NCI Dictionary of Cancer Terms. Retrieved November 20, 2025, from <https://www.cancer.gov/publications/dictionaries/cancer-terms/def/socioeconomic-status>

Housing, Infrastructure and Communities Canada. (2025, September 12). Investing in Canada Plan – Building a Better Canada. Retrieved November 20, 2025, from <https://housing-infrastructure.canada.ca/plan/about-invest-apropos-eng.html>

Becker, B., & Kohavi, R. (1996). Adult [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5XW20>

Yassin, S., Petit, G., & Abraham, Y. (2024, July 18). The troubling rise of income and wealth inequality in Canada. Policy Options. Institute for Research on Public Policy. <https://policyoptions.irpp.org/2024/07/income-wealth-inequality/>

Chen, J. (n.d.). Feature Significance Analysis of the US Adult Income Dataset (TR 1869) [Technical report, University of Wisconsin–Madison]. UW–Madison Institutional Repository. <https://minds.wisconsin.edu/bitstream/handle/1793/82299/TR1869%20Junda%20Chen%203.pdf?sequence=1&is>

[ ]: