Knowledge Discovery in Databases (KDD) for World Adult Income Analysis

Suresh Ravuri (Generated with The Help Of GPT)

September 26, 2023

ABSTRACT

This paper presents a comprehensive exploration of the Knowledge Discovery in Databases (KDD) process applied to the "WAI.data" dataset. The dataset contains attributes such as age, workclass, education, and income. Using the PyCaret library, we streamlined the process of data preprocessing, transformation, model training, and evaluation to predict income levels.

1. INTRODUCTION

The KDD process is a systematic approach to uncovering valuable insights from large volumes of data. This research leverages the KDD methodology to analyze the adult income dataset, aiming to predict income levels based on various attributes.

2. KNOWLEDGE DISCOVERY IN DATABASE (KDD) METHODOLOGY

2.1 Data Selection:

The "WAI.data" dataset was chosen, encompassing attributes like age, workclass, education, and more. The primary goal was to predict the income level based on these attributes.

In this section, we import necessary libraries for data handling and visualization. We also leverage Google Colab's built-in files.upload() function to facilitate the seamless upload of our dataset.

In [2]:
```
from google.colab import files
```

In [73]:
```
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving adult.data to adult.data

In [74]:
```
#The uploaded file's name is 'adult.data'
column_names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship',
data = pd.read_csv('adult.data', names=column_names, sep='\s*,\s*', engine='python')
```

2.2 Data Preprocessing:

Data in the real world is messy. In the preprocessing step, we handle missing values, outliers, and possibly noisy data. This might involve imputing missing data, filtering out outliers, or smoothing noisy data. Preprocessing ensures that our data is of high quality and ready for the next steps.

Real-world data is often riddled with missing values, outliers, and noise. Our preprocessing involved:

- Replacing '?' characters with NaN values.
- Calculating the number of missing values for each attribute.
- Descriptive statistics to understand the dataset's distribution.

In [75]:
```
data.head()
```

Out[75]:

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	in
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	

```
In [76]: data.replace('?', np.NaN, inplace=True)

In [77]: data.isnull().sum()

Out[77]: age                0
         workclass       1836
         fnlwgt             0
         education          0
         education-num      0
         marital-status     0
         occupation      1843
         relationship       0
         race               0
         sex                0
         capital-gain       0
         capital-loss       0
         hours-per-week     0
         native-country   583
         income             0
         dtype: int64
```

2.3 Data Transformation:

Objective: Convert data into a suitable format or structure for analysis.

In the transformation phase, we might need to normalize or scale our data, especially if we're dealing with features that have different units or scales. We might also generate new features, aggregate data, or even decompose existing features. The goal is to structure our data in a way that enhances the capability of our analytical tools.

```
In [78]: data.describe()
```

Out[78]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

2.4 Data Mining:

Objective: Apply data mining techniques to extract patterns.

This is the core step of the KDD process. Here, we use various data mining algorithms to find patterns, associations, anomalies, or structures in our data.

Depending on our goal, this might involve clustering, classification, regression, or even association rule mining.

In [80]:
```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.8718	0.9270	0.6635	0.7726	0.7135	0.6316	0.6349	1.7030
catboost	CatBoost Classifier	0.8706	0.9281	0.6582	0.7714	0.7100	0.6274	0.6309	13.8250
xgboost	Extreme Gradient Boosting	0.8671	0.9242	0.6551	0.7602	0.7036	0.6186	0.6215	7.9100
gbc	Gradient Boosting Classifier	0.8652	0.9213	0.6079	0.7845	0.6847	0.6008	0.6089	5.5790
ada	Ada Boost Classifier	0.8598	0.9170	0.6127	0.7592	0.6779	0.5896	0.5953	1.7880
rf	Random Forest Classifier	0.8525	0.9035	0.6263	0.7249	0.6715	0.5772	0.5801	4.0330
lda	Linear Discriminant Analysis	0.8396	0.8933	0.5602	0.7127	0.6272	0.5269	0.5332	0.8600
ridge	Ridge Classifier	0.8389	0.0000	0.5014	0.7468	0.5997	0.5038	0.5196	0.5940
et	Extra Trees Classifier	0.8315	0.8794	0.6048	0.6658	0.6335	0.5245	0.5257	4.6480
dt	Decision Tree Classifier	0.8142	0.7509	0.6287	0.6115	0.6197	0.4969	0.4972	0.6860
lr	Logistic Regression	0.7997	0.6325	0.2977	0.7042	0.4086	0.3144	0.3605	2.0880
nb	Naive Bayes	0.7923	0.8307	0.3017	0.6480	0.4113	0.3051	0.3381	0.9290
knn	K Neighbors Classifier	0.7713	0.6523	0.3017	0.5460	0.3885	0.2620	0.2795	1.9600
dummy	Dummy Classifier	0.7592	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6820
svm	SVM - Linear Kernel	0.6787	0.0000	0.3631	0.6409	0.3268	0.1860	0.2423	2.2690
qda	Quadratic Discriminant Analysis	0.4555	0.6923	0.9534	0.3043	0.4599	0.1484	0.2554	0.8290

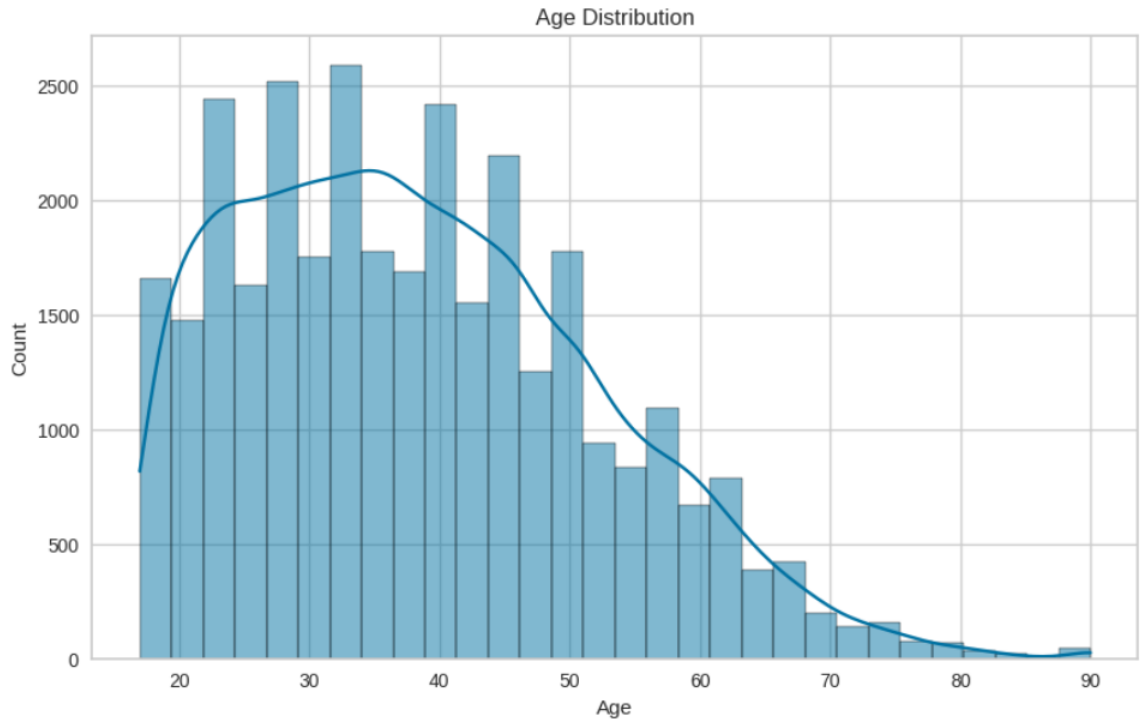
```
Processing:   0%|          | 0/69 [00:00<?, ?it/s]
```

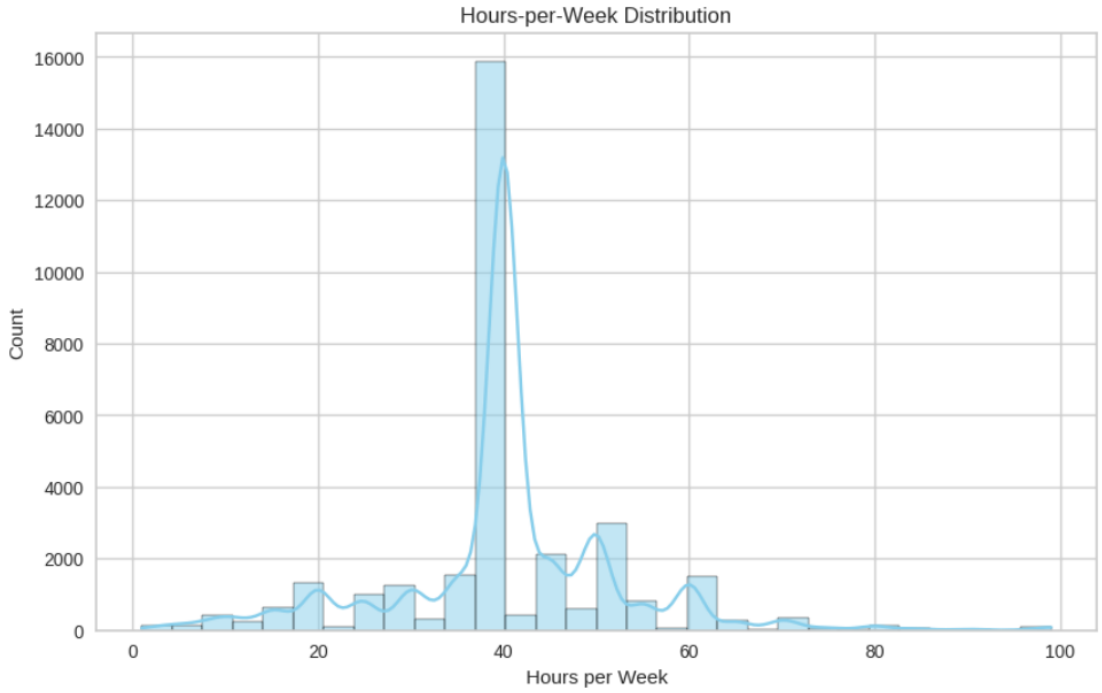
2.5 Visualization:

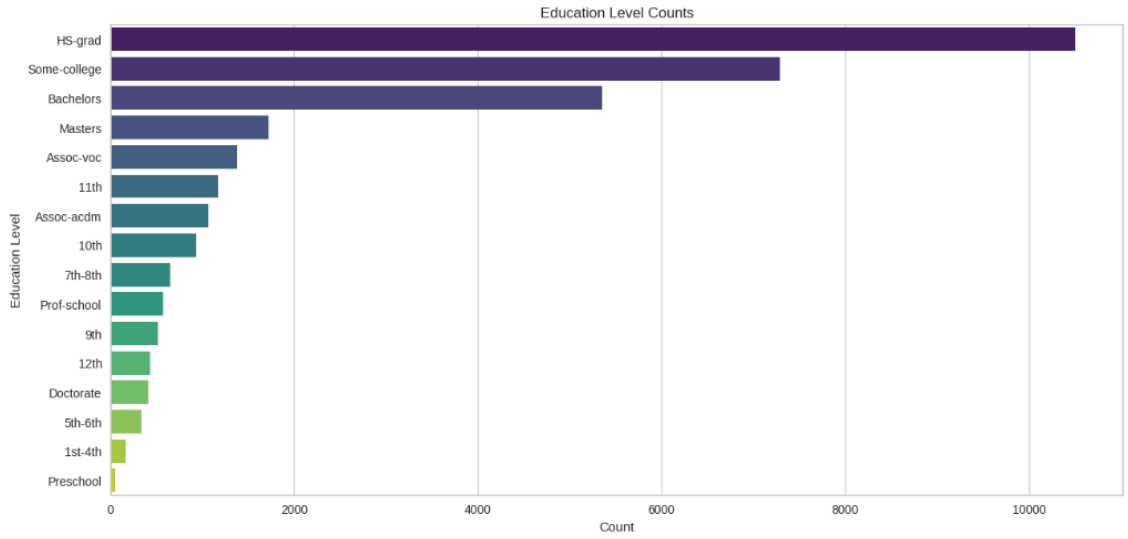
Several visualizations provided insights into the data:

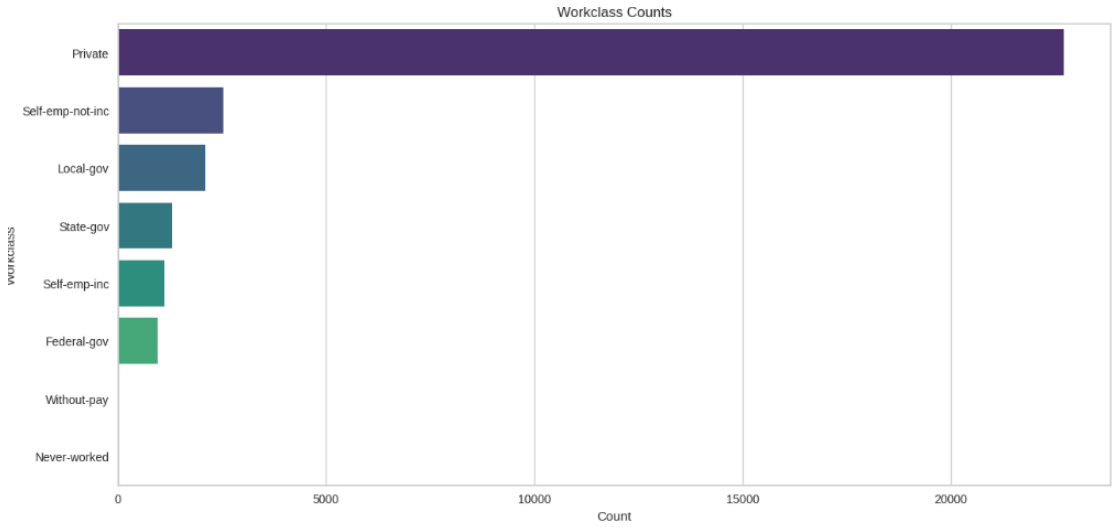
- Age distribution histogram.
- Hours-per-week distribution histogram.
- Count plots for education levels, workclass, and marital status.

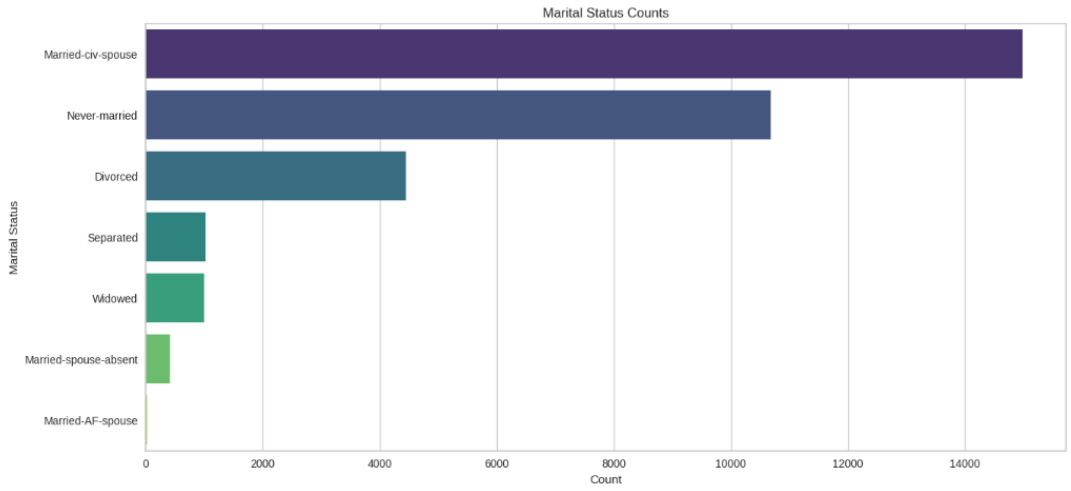
In [81]:
```
# Histogram for age distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['age'], bins=30, kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()

# Histogram for hours-per-week distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['hours-per-week'], bins=30, kde=True, color='skyblue')
plt.title('Hours-per-Week Distribution')
plt.xlabel('Hours per Week')
plt.ylabel('Count')
plt.show()
```


Age Distribution
Count
Age
0
500
1000
1500
2000
2500
20
30
40
50
60
70
80
90


Hours-per-Week Distribution
Count
Hours per Week
0
2000
4000
6000
8000
10000
12000
14000
16000
0
20
40
60
80
100


Education Level Counts
HS-grad
Some-college
Bachelors
Masters
Assoc-voc
11th
Assoc-acdm
10th
7th-8th
Prof-school
9th
12th
Doctorate
5th-6th
1st-4th
Preschool
Education Level
0
2000
4000
6000
8000
10000
Count


Workclass Counts
Private
Self-emp-not-inc
Local-gov
State-gov
Self-emp-inc
Federal-gov
Without-pay
Never-worked
workclass
0
5000
10000
15000
20000
Count


Marital Status Counts
Married-civ-spouse
Never-married
Divorced
Separated
Widowed
Married-spouse-absent
Married-AF-spouse
Marital Status
0
2000
4000
6000
8000
10000
12000
14000
Count

3. Interpretation/Evaluation

Objective: Interpret and evaluate the mined patterns.

Once we've mined some patterns or models, it's crucial to interpret them in the context of our domain or problem. This might involve visualizations, statistical tests, or even domain-specific knowledge. We also need to evaluate our findings: Are the patterns valid? Are they novel? Are they actionable? If a pattern doesn't meet these criteria, it might not be useful, regardless of how statistically significant it is

In [83]:
```
evaluate_model(best_model)
```

```
interactive(children=(ToggleButtons(description='Plot Type:', icons=('',), options=(('Pipeline Plot', 'pipelin…
```

In [84]:
```
final_model = finalize_model(best_model)
save_model(final_model, 'final_model_adult_income')
```

```
Transformation Pipeline and Model Successfully Saved
```

Out[84]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, include=None,
                                                transformer=LabelEncoder())),
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['age', 'fnlwgt', 'education-num',
                                              'capital-gain', 'capital-loss',
                                              'hours-per-week'],
                                     transformer=SimpleImputer(add_indicator=False,
                                                               copy=True,
                                                               fill_...
                  LGBMClassifier(boosting_type='gbdt', class_weight=None,
                                 colsample_bytree=1.0, importance_type='split',
                                 learning_rate=0.1, max_depth=-1,
                                 min_child_samples=20, min_child_weight=0.001,
                                 min_split_gain=0.0, n_estimators=100, n_jobs=-1,
                                 num_leaves=31, objective=None, random_state=123,
                                 reg_alpha=0.0, reg_lambda=0.0, subsample=1.0,
                                 subsample_for_bin=200000, subsample_freq=0))],
          verbose=False),
 'final_model_adult_income.pkl')
```

4. Discussion

The best model provided satisfactory results for predicting income levels. Visualizations played a pivotal role in understanding the data's distribution and the relationship between different attributes and the target variable. The PyCaret library significantly streamlined the machine learning process, allowing for efficient model comparison and evaluation.

4. Conclusion

This research showcased the application of the KDD process to the adult income dataset. By following structured steps, from data selection to interpretation, we derived actionable insights and a predictive model for income levels. Future studies could delve deeper into feature engineering and hyperparameter tuning to enhance model performance.

References

World Adult Income Dataset - https://medium.com/@sureshravuri07/analyzing-the-world-adult-income-dataset-kdd-analysis-4020b8008114

PyCaret: Official Documentation