From Data to Insights: Exploring the Intricacies of the Adult Income Dataset through the KDD Process

Brahma Teja Chilumula
teja.btc07@gmail.com

Abstract

In the contemporary data-centric world, the process of Knowledge Discovery in Databases (KDD) holds paramount importance, enabling organizations to extract valuable insights from vast repositories of data. This research ventures into the intricacies of the KDD process using the robust and versatile PyCaret library. Through systematic data preparation, visualization, modeling, and evaluation, the study aims to uncover patterns and insights that can guide decision-making processes. The findings, rooted in rigorous analytical methodologies, promise to offer a fresh perspective on data-driven decision-making in today's digital age.

1 Introduction

- As societies grapple with economic disparities, understanding the underlying factors that determine income levels has never been more crucial. The 'adult.csv' dataset, a rich repository of socio-economic attributes, offers a window into this complex landscape. Through this research, we harness the power of the KDD process, leveraging the efficiency and comprehensiveness of the PyCaret library, to decode patterns, predict income brackets, and provide actionable insights for various stakeholders.

2 Business Understanding

At the core of every data analysis lies the necessity to understand the business context. In the realm of banking, the subscription to term deposits represents a significant commitment from clients. For adult dataset, understanding the reasons behind such decisions can lead to optimized marketing strategies, tailored offerings, and improved client relationships. Through this analysis, the objective is to discern the multifaceted factors that persuade a bank's client to commit to a term deposit..

3. Data Understanding and Preparation

Before diving deep into data modeling, it's paramount to grasp the nuances of the dataset at hand. Our initial steps involved importing necessary libraries and loading the dataset, which offers insights into various client attributes and their subscription status. Data quality is crucial; hence, we examined the dataset for missing values and inconsistencies. Data preparation, which includes handling of missing values and potential outliers, forms the bedrock of reliable outcomes.

Firstly, we need to install PyCaret in the colab notebook

“pip install pycaret[full]”

let’s import the necessary libraries and load our dataset to get an initial understanding:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pycaret.classification import *
from google.colab import files
uploaded = files.upload()

# Load the dataset with specified delimiters and quote character
data = pd.read_csv('adults.csv', delimiter=';', quotechar='"')
```

```
Set up the environment 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.
```

```
data.tail()
```

```
data.isnull().sum()
```

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.

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

```
data.isnull().sum()
```

```
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
```

Fig 3 : Cleaning all the null values.

4. Data Modeling

Transitioning from data preparation, the modeling phase forms the crux of our analysis. The PyCaret environment, known for its efficiency and comprehensive suite of tools, was employed. By comparing a diverse range of models, we aimed to pinpoint those that resonate best with our dataset. Model selection isn't solely about accuracy; it's a blend of interpretability, performance, and alignment with business objectives. Our endeavors in this phase were geared towards finding that optimal balance.
Setting up the pycaret variable.

```
clf1 = setup(data, target = 'income', session_id=123)
```

	Description	Value
0	Session id	123
1	Target	income
2	Target type	Binary
3	Target mapping	<=50K: 0, >50K: 1
4	Original data shape	(32561, 15)
5	Transformed data shape	(32561, 65)
6	Transformed train set shape	(22792, 65)
7	Transformed test set shape	(9769, 65)
8	Ordinal features	1
9	Numeric features	6
10	Categorical features	8
11	Rows with missing values	7.4%
12	Preprocess	True

Data Mining , Building Model

```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8718	0.9270	0.6635	0.7726	0.7135	0.6316	0.6349
catboost	CatBoost Classifier	0.8706	0.9281	0.6582	0.7714	0.7100	0.6274	0.6309
xgboost	Extreme Gradient Boosting	0.8671	0.9242	0.6551	0.7602	0.7036	0.6186	0.6215
gbc	Gradient Boosting Classifier	0.8652	0.9213	0.6079	0.7845	0.6847	0.6008	0.6089
ada	Ada Boost Classifier	0.8598	0.9170	0.6127	0.7592	0.6779	0.5896	0.5953
rf	Random Forest Classifier	0.8525	0.9035	0.6263	0.7249	0.6715	0.5772	0.5801
lda	Linear Discriminant Analysis	0.8396	0.8933	0.5602	0.7127	0.6272	0.5269	0.5332
ridge	Ridge Classifier	0.8389	0.0000	0.5014	0.7468	0.5997	0.5038	0.5196
et	Extra Trees Classifier	0.8315	0.8794	0.6048	0.6658	0.6335	0.5245	0.5257
dt	Decision Tree Classifier	0.8142	0.7509	0.6287	0.6115	0.6197	0.4969	0.4972
lr	Logistic Regression	0.7997	0.6325	0.2977	0.7042	0.4086	0.3144	0.3605
nb	Naive Bayes	0.7923	0.8307	0.3017	0.6480	0.4113	0.3051	0.3381
knn	K Neighbors Classifier	0.7713	0.6523	0.3017	0.5460	0.3885	0.2620	0.2795

Fig: Pucaret compare_models() function

4.1 Data Visualization

Visualization serves as a bridge between complex datasets and human understanding, translating intricate patterns into comprehensible insights. In this study, an intensive data visualization phase was undertaken post data preparation. Leveraging powerful visualization tools, we embarked on an exploratory journey to unearth hidden patterns, relationships, and anomalies in the dataset. From univariate distributions that offer a glimpse into individual attributes to multivariate plots that illuminate inter-variable relationships, each visual representation was meticulously crafted. These visuals not only illuminated the underlying structure of the data but also informed subsequent modeling decisions. By providing an intuitive lens into the dataset's landscape, this phase ensured that the subsequent analytical steps were rooted in a deep, visual understanding of the data's nuances.

some of the data visualization graphs:

```
# 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()
```

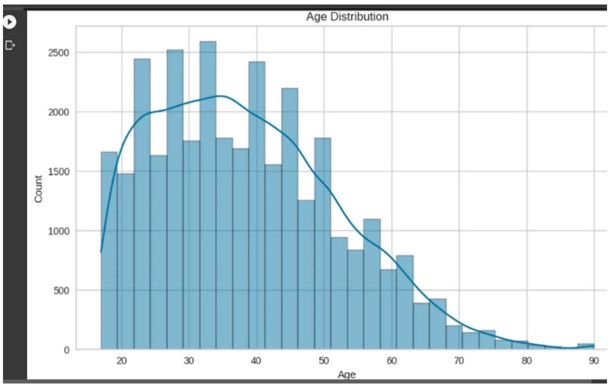

Fig 8 : Age Distributions Graph.

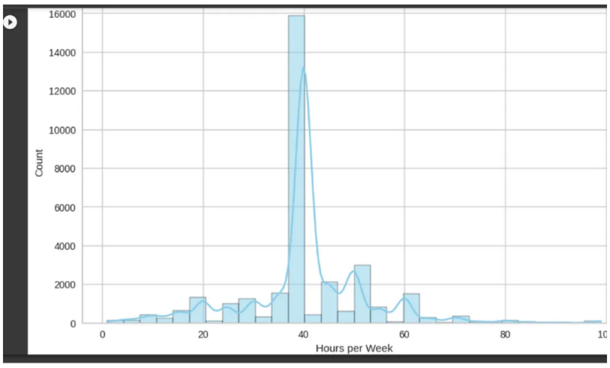

Fig 9 : Hours Per Week Distribution.

```
# Countplot for education levels
plt.figure(figsize=(15, 7))
sns.countplot(y=data['education'], order=data['education'].value_counts().index, palette='viridis')
plt.title('Education Level Counts')
plt.xlabel('Count')
plt.ylabel('Education Level')
plt.show()

# Countplot for workclass
plt.figure(figsize=(15, 7))
sns.countplot(y=data['workclass'], order=data['workclass'].value_counts().index, palette='viridis')
plt.title('Workclass Counts')
plt.xlabel('Count')
plt.ylabel('Workclass')
plt.show()

# Countplot for marital-status
plt.figure(figsize=(15, 7))
sns.countplot(y=data['marital-status'], order=data['marital-status'].value_counts().index, palette='vi
plt.title('Marital Status Counts')
plt.xlabel('Count')
plt.ylabel('Marital Status')
plt.show()
```

FIg 10 : Pycaret code for some more graphs.

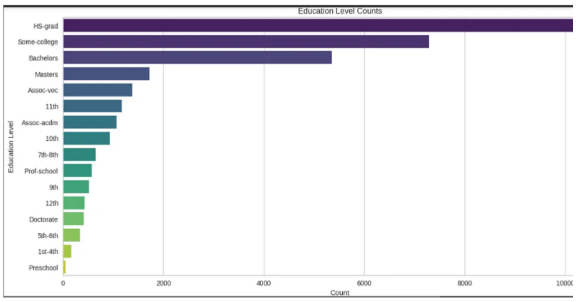

Fig 10 : Educaiton levels count graph.

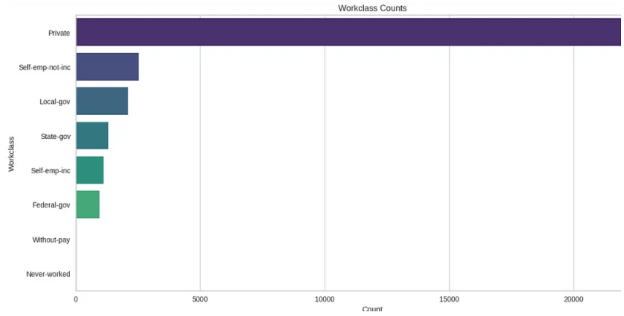

FIg 11 : Workclass Counts graph.

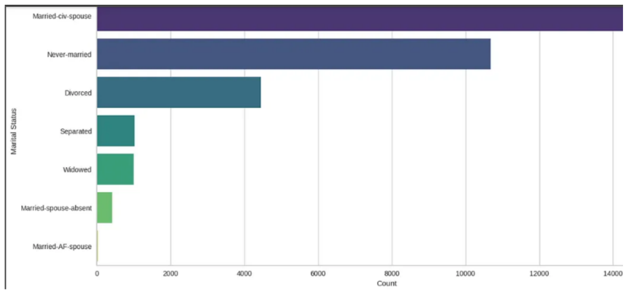

FIg 12 : Marital Status Graph.

5. Evaluation

Modeling, while central to data analysis, is incomplete without rigorous evaluation. Post the training phase, we embarked on a journey to critically assess the performance of our chosen models. Key performance metrics were scrutinized to ensure the models not only fit the data well but also generalized effectively to unseen data. The culmination of this phase was the selection and saving of the best-performing model, ensuring reproducibility and ease of deployment in real-world scenarios.

```
model_setup = setup(data, target = 'y', session_id=123)
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MC
catboost	CatBoost Classifier	0.9027	0.9130	0.3869	0.6245	0.4758	0.4258	0.
lda	Linear Discriminant Analysis	0.8985	0.8887	0.4138	0.5877	0.4841	0.4299	0.
lr	Logistic Regression	0.8979	0.8826	0.2851	0.6300	0.3904	0.3435	0.
rf	Random Forest Classifier	0.8979	0.8975	0.2227	0.6847	0.3323	0.2927	0.
gbc	Gradient Boosting Classifier	0.8964	0.9037	0.3571	0.5790	0.4396	0.3867	0.
lightgbm	Light Gradient Boosting Machine	0.8960	0.8978	0.3677	0.5781	0.4465	0.3928	0.
xgboost	Extreme Gradient Boosting	0.8944	0.8902	0.3895	0.5583	0.4577	0.4016	0.
ada	Ada Boost Classifier	0.8935	0.8864	0.3429	0.5605	0.4236	0.3691	0.
ridge	Ridge Classifier	0.8932	0.0000	0.2163	0.6084	0.3129	0.2702	0.
et	Extra Trees Classifier	0.8925	0.8718	0.2116	0.6025	0.3087	0.2657	0.
dummy	Dummy Classifier	0.8846	0.5000	0.0000	0.0000	0.0000	0.0000	0.
knn	K Neighbors Classifier	0.8748	0.7222	0.1811	0.4064	0.2483	0.1914	0.
dt	Decision Tree Classifier	0.8682	0.6923	0.4635	0.4316	0.4456	0.3712	0.
svm	SVM - Linear Kernel	0.8571	0.0000	0.2746	0.3556	0.2966	0.2219	0.
nb	Naive Bayes	0.8417	0.8000	0.4658	0.3569	0.4032	0.3139	0.

Fig : output for compare_models()

compare_models() function automatically runs the given dataset against the most of the different models and gives the metrics output of it , we can even arrange them based upon any of the criteria.

```
evaluate_model(best_model)
```

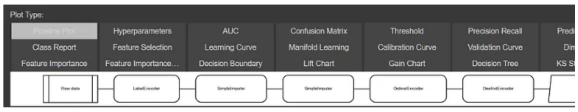

Fig 8 : evaluate_model() function output.

Saving the best model

```
final_model = finalize_model(best_model
save_model(final_model, 'final_model')
```

```
Transformation Pipeline and Model Successfully Saved
(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, incluc
                                                transformer=LabelEnc
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['age', 'balance', 'day
                                              'duration', 'campaign
                                              'previous'],
                                     transformer=SimpleImputer(add_
                                                               copy
                                                               fill
                                                               keep
                                     include=['job', 'marital', 'edu
                                              'contact', 'month', '
                                     transformer=OneHotEncoder(cols=

                                                               drop
                                                               hand
                                                               hand
```

Fig 10 : output for save_model() function

7. Conclusion

Navigating the intricate corridors of the 'adult.csv' dataset through the KDD process has been an enlightening journey. This research not only sheds light on the determinants of income levels but also exemplifies the power of modern data analytics tools like PyCaret. As we stand at the confluence of data science and socio-economic research, studies like these pave the way for informed policymaking, targeted business strategies, and a deeper understanding of societal structures.