PHASE 3 PROJECT

LESLEY KAMAMO

Overview

Stepping into Billionaire Territory

This project requires the use Classification Model to generate insights for a given Agency.

This agency provides all sorts of social support in the different industries, mostly volunteer work. In a bid to increase their profits for the next financial year, their have been looking through different datasets that will enable them make a choice of industriy investment.

Business Understanding

BUSINESS PROBLEM

BUSINESS OBJECTIVES

The objectives of this project based on the dataset chosen is to find out:

1. The Wealth Status of Billionaires (whether self-made or not)

2. What industry/sources is more inclined to produce billionaires?

3. Demographic analysis of billionaires (age, gender, country)

4. Provide classification to the wealth status of billionaires based on the features.

5. Provide insights into which industry are likely to produce billionaires in future (logistic regression)

Data Understanding

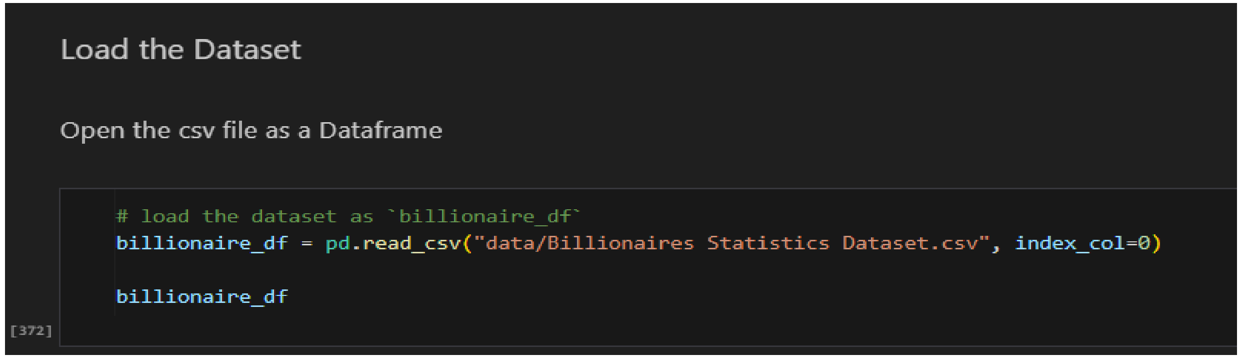
The dataset represents historical data on billionaires for recent past years, hence this data will be modified for the purpose of the analysis

The data is contained in a CSV file:

1. `Billionaires Statistics Dataset.csv`: each record represents rank, finalWorth, category, personName, age, country,source, selfMade,status,gender,birthDate,title,residenceStateRegion,birthYear, tax_revenue_country_country,total_tax_rate_country,population_country among other fields.

Data Preparation

1. Loading the Dataset
2. Handling Missing Values
3. Describing the Data

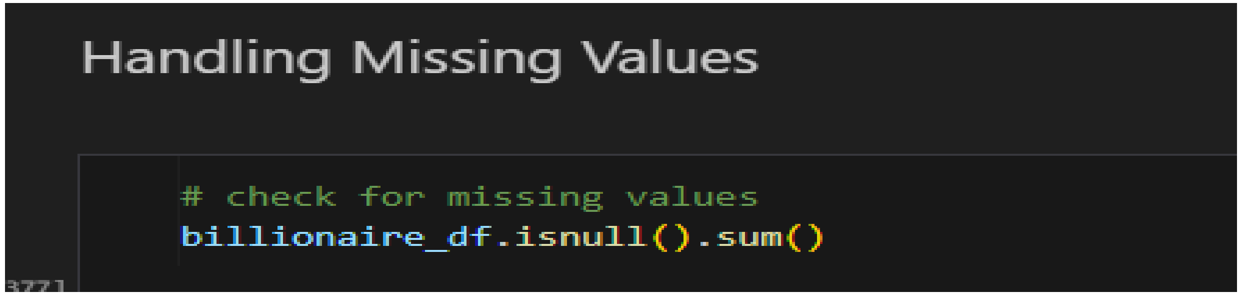


Load the Dataset

Open the csv file as a Dataframe

```
# load the dataset as `billionaire_df`
billionaire_df = pd.read_csv("data/Billionaires Statistics Dataset.csv", index_col=0)

billionaire_df
```

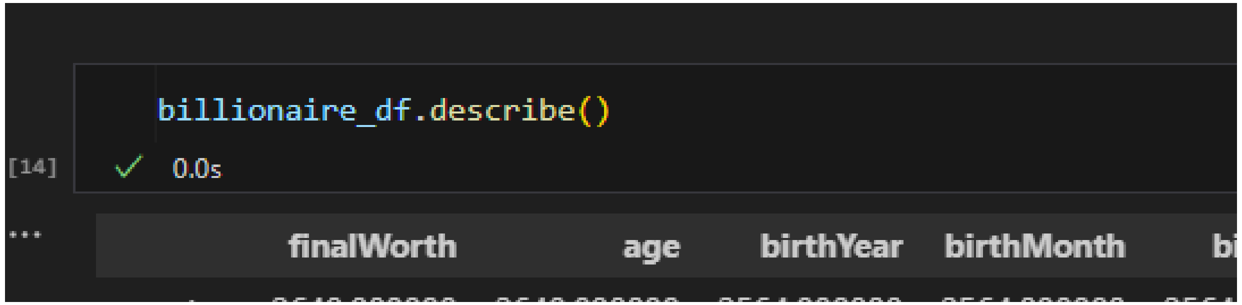
[372]



Handling Missing Values

```
# check for missing values
billionaire_df.isnull().sum()
```

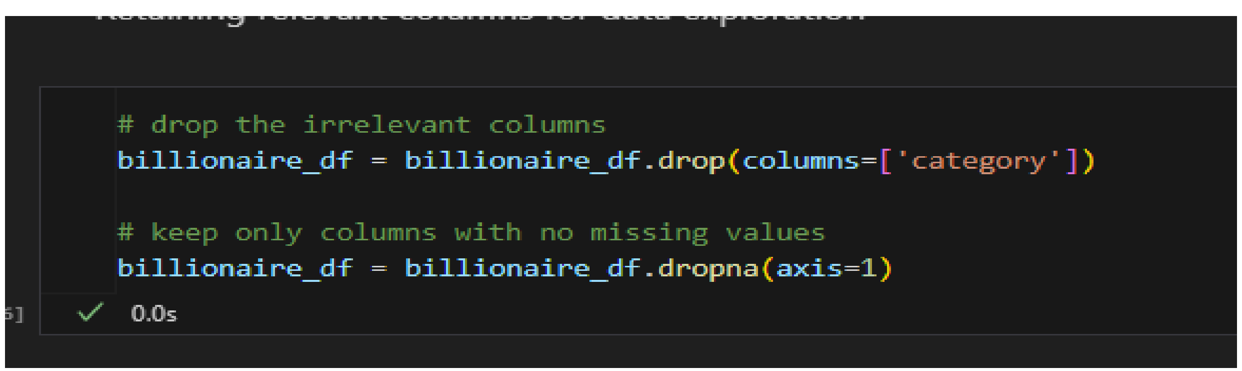
[377]



```
billionaire_df.describe()
```

[14] ✓ 0.0s

```
...        finalWorth         age     birthYear   birthMonth     bi
```



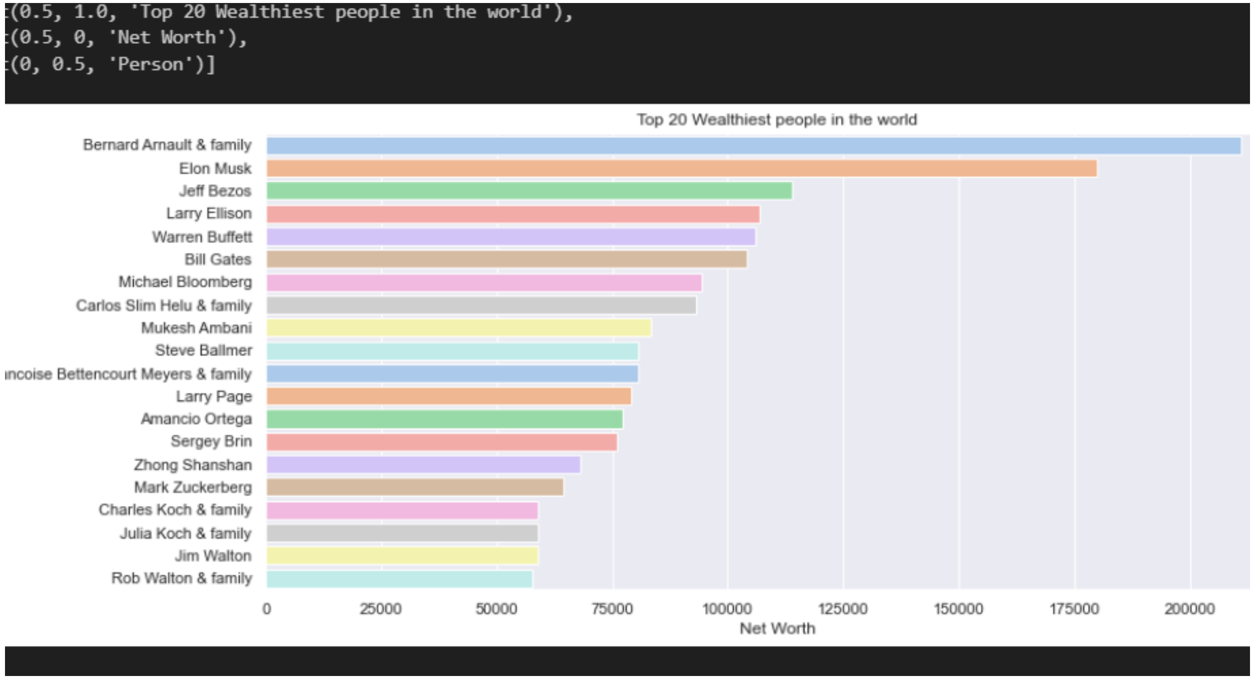
Retaining relevant columns for data exploration

```
# drop the irrelevant columns
billionaire_df = billionaire_df.drop(columns=['category'])

# keep only columns with no missing values
billionaire_df = billionaire_df.dropna(axis=1)
```

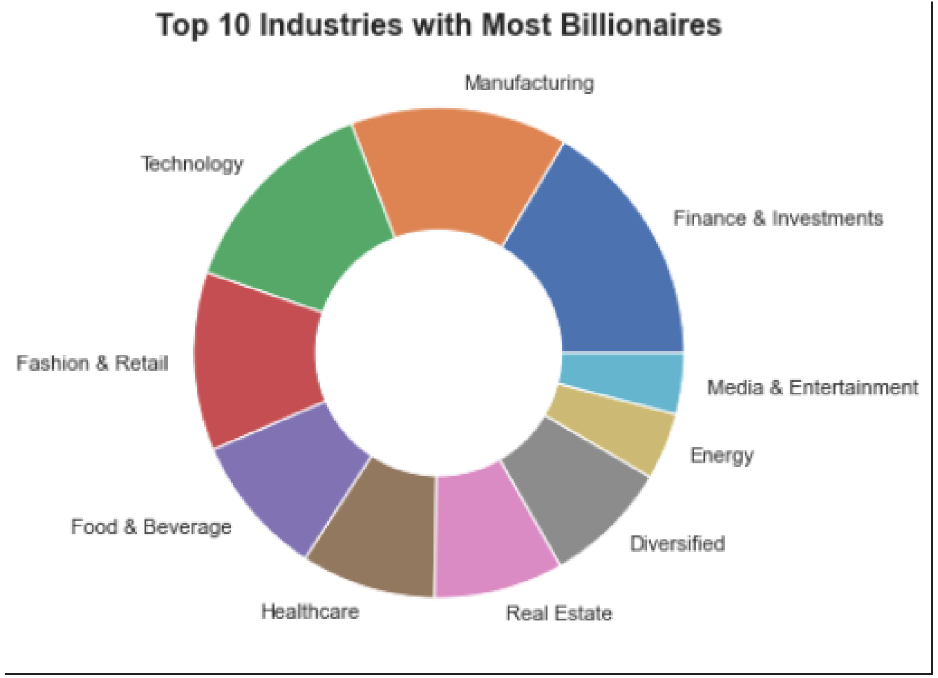
6] ✓ 0.0s

Exploratory Data Analysis

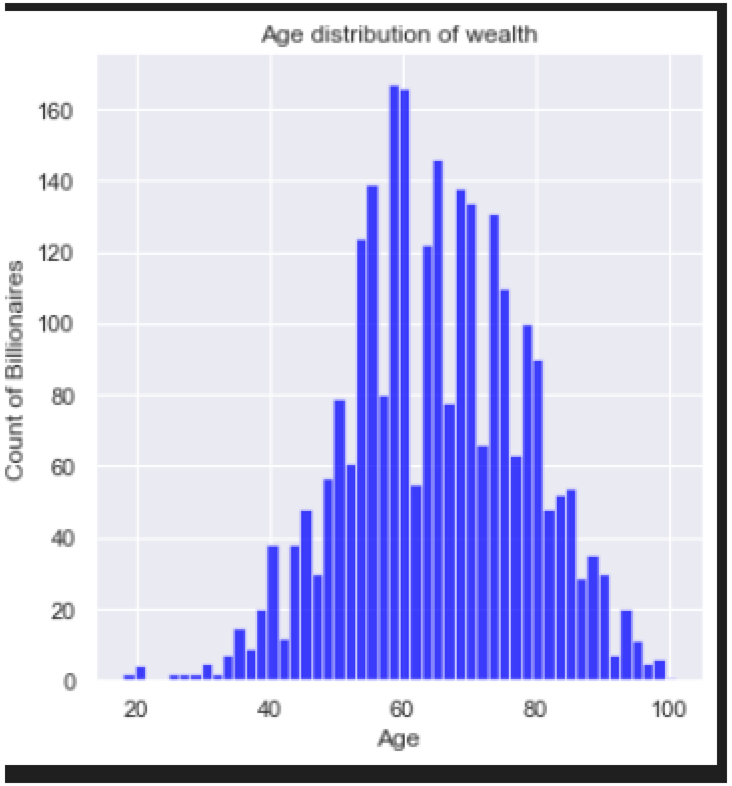
1. Univariate Analysis
2. Bivariate Analysis

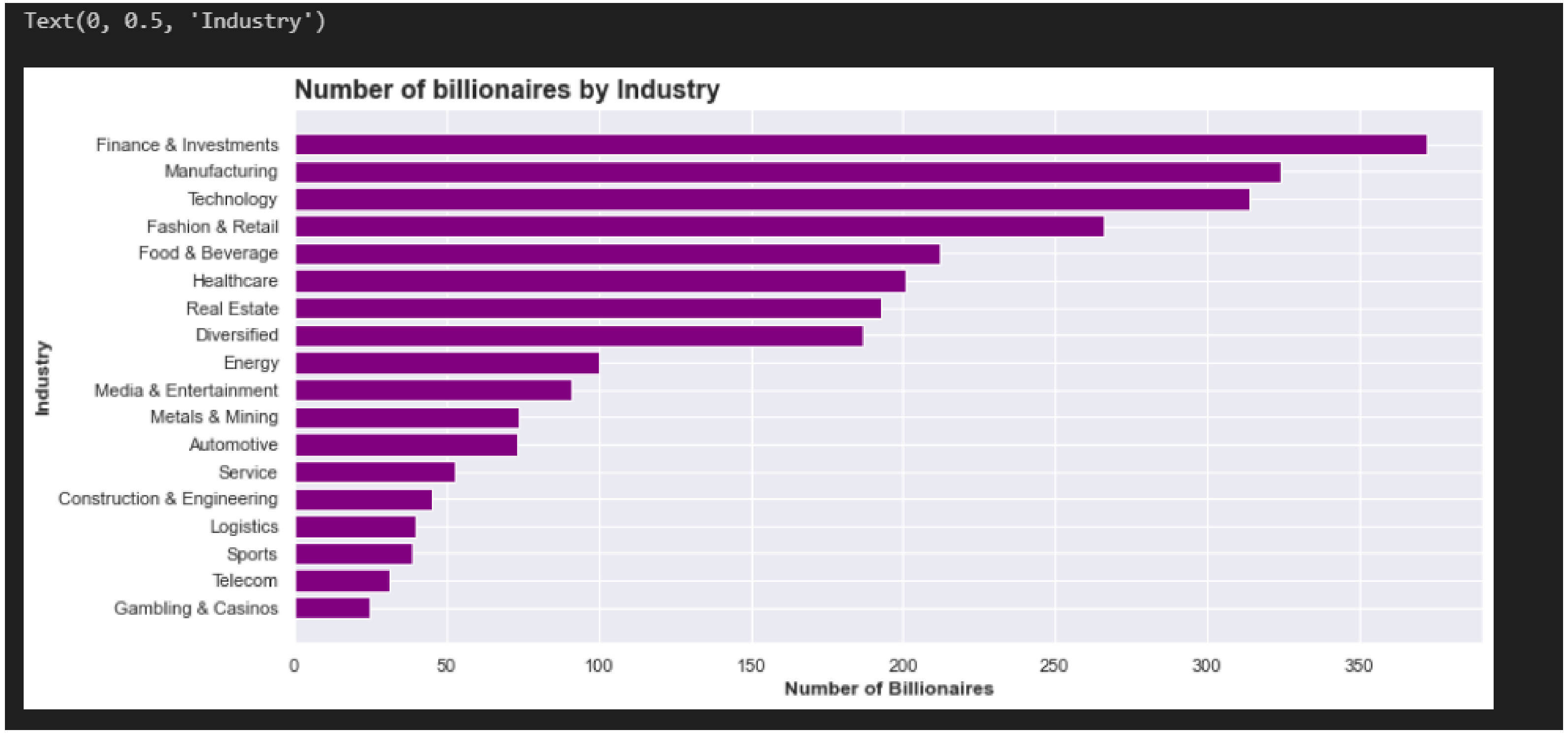


```
(0.5, 1.0, 'Top 20 Wealthiest people in the world'),
(0.5, 0, 'Net Worth'),
(0, 0.5, 'Person')]
```

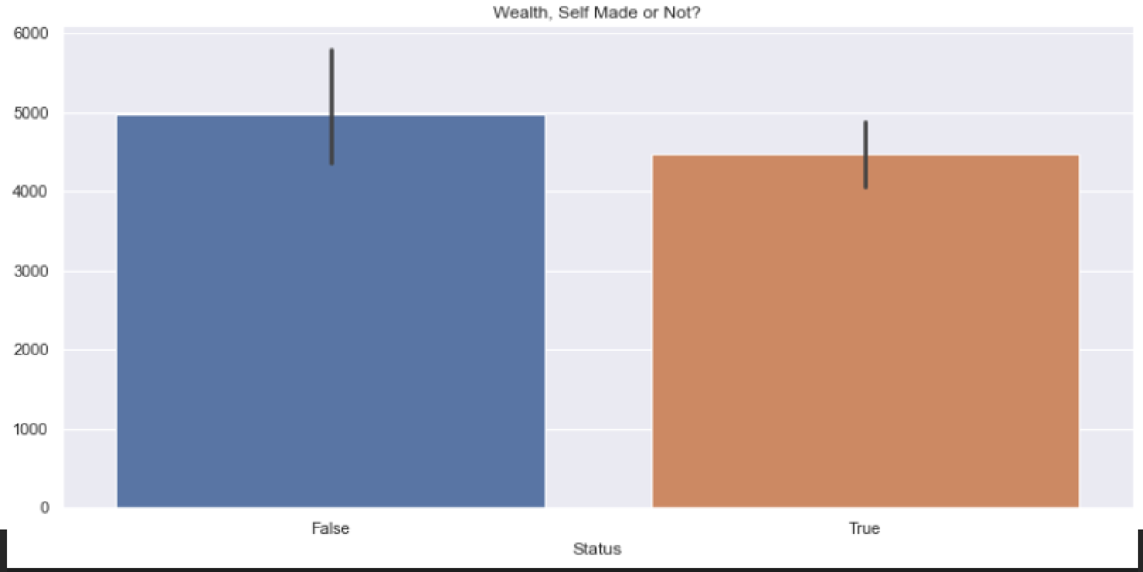


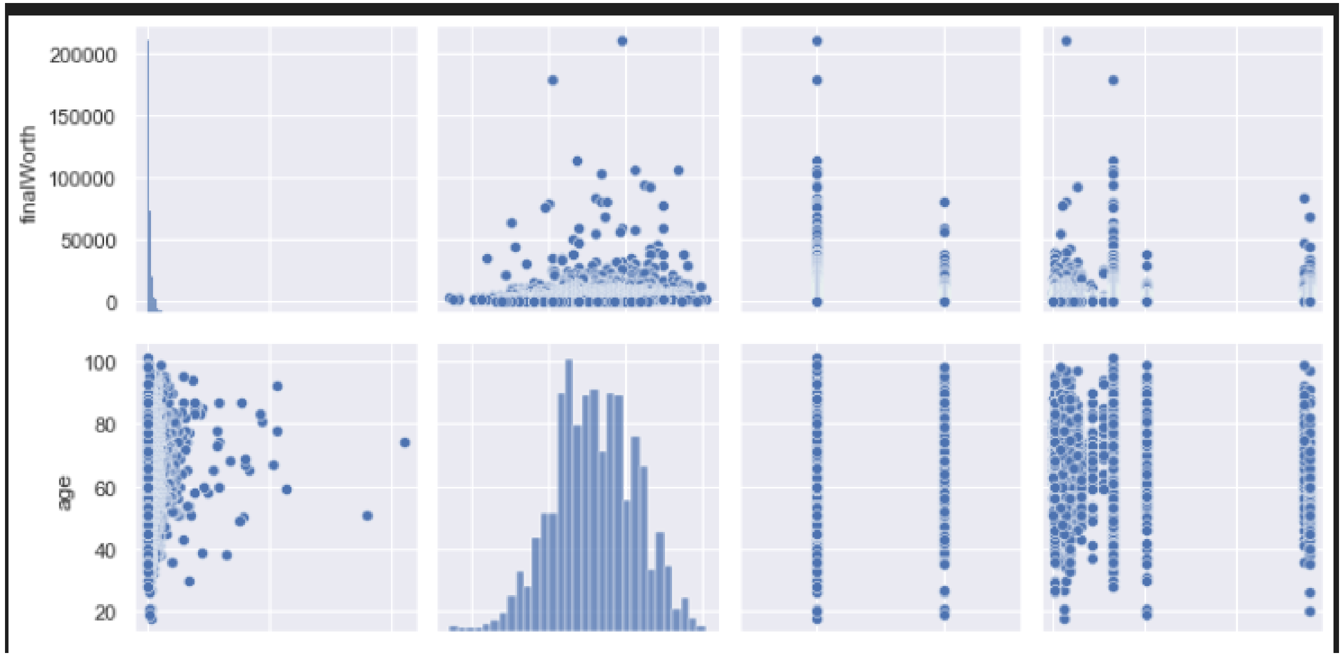
Top 10 Industries with Most Billionaires



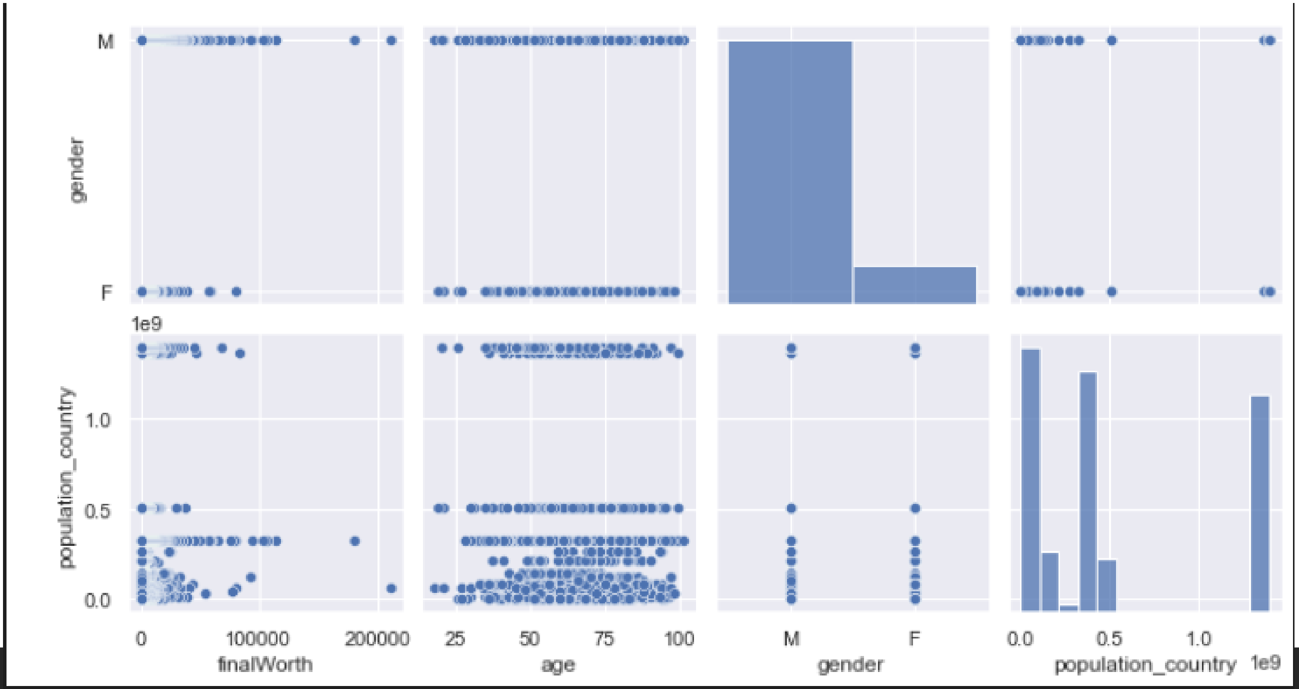


```
Text(0, 0.5, 'Industry')
```



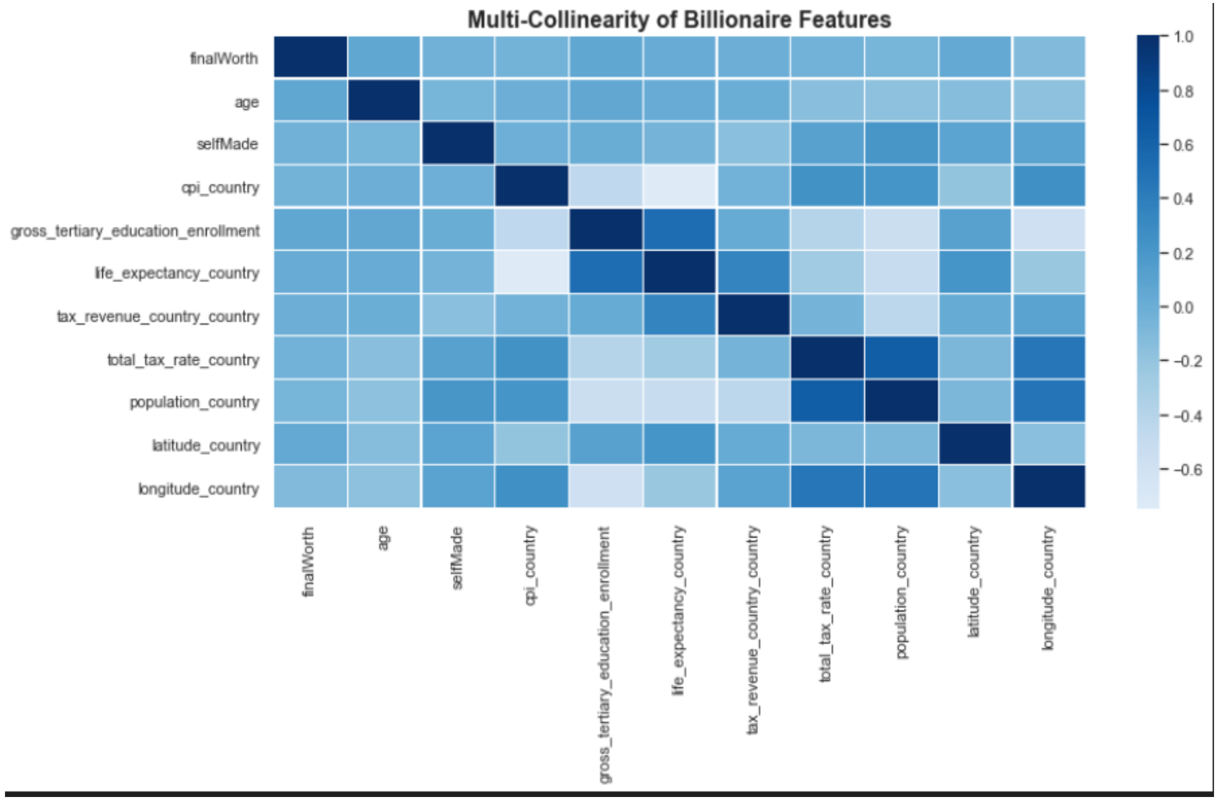


finalWorth
200000
150000
100000
50000
0
age
100
80
60
40
20

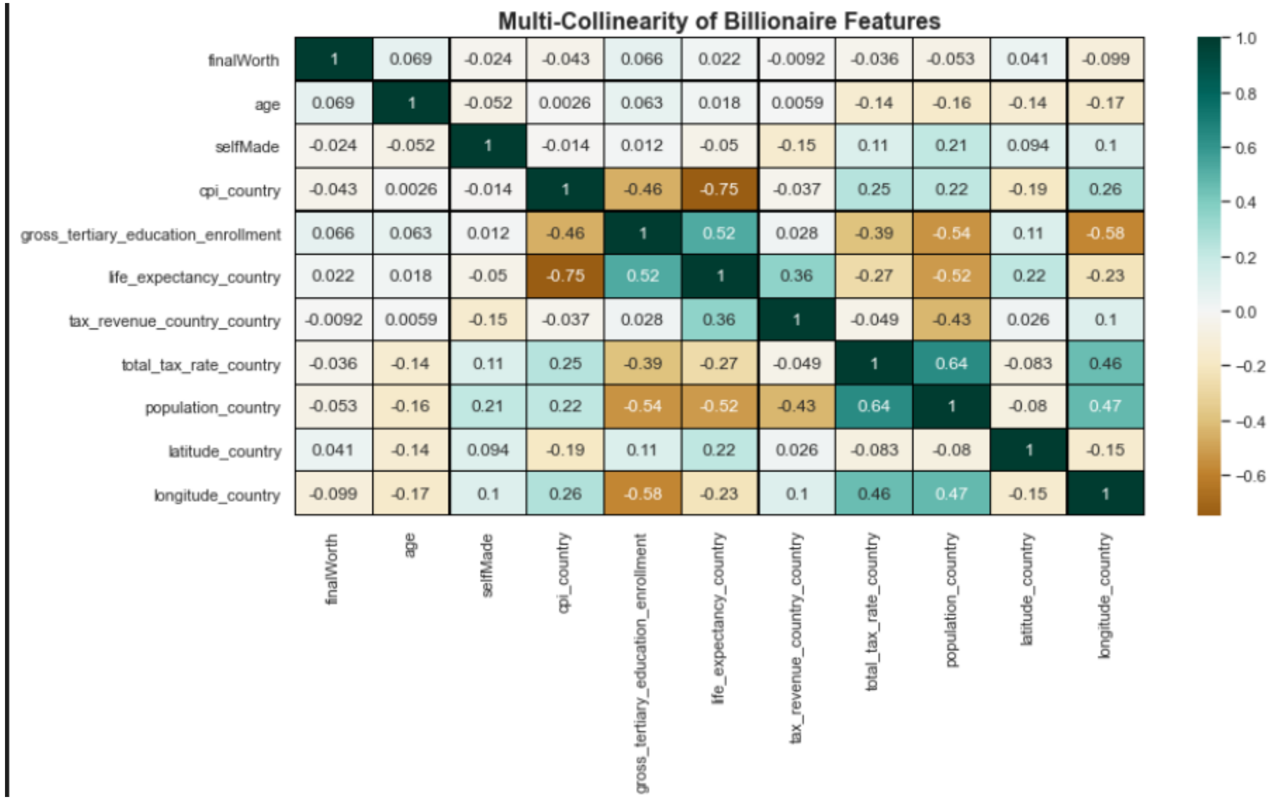


gender
M
F
1e9
population_country
1.0
0.5
0.0
0
100000
200000
finalWorth
25
50
75
100
age
M
F
gender
0.0
0.5
1.0
population_country
1e9

Multi-Collinearity



Multi-Collinearity of Billionaire Features



Multi-Collinearity of Billionaire Features

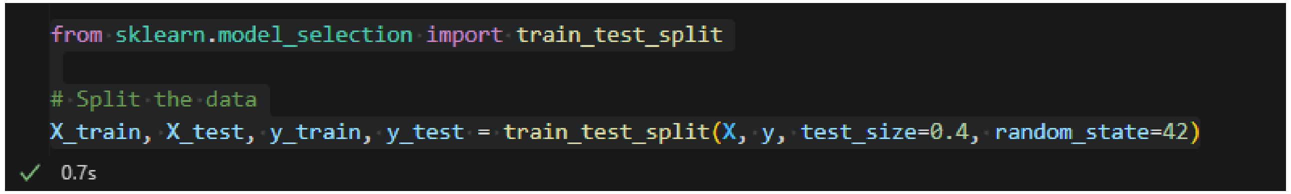
	finalWorth	age	selfMade	cpi_country	gross_tertiary_education_enrollment	life_expectancy_country	tax_revenue_country_country	total_tax_rate_country	population_country	latitude_country	longitude_country
finalWorth	1	0.069	-0.024	-0.043	0.066	0.022	-0.0092	-0.036	-0.053	0.041	-0.099
age	0.069	1	-0.052	0.0026	0.063	0.018	0.0059	-0.14	-0.16	-0.14	-0.17
selfMade	-0.024	-0.052	1	-0.014	0.012	-0.05	-0.15	0.11	0.21	0.094	0.1
cpi_country	-0.043	0.0026	-0.014	1	-0.46	-0.75	-0.037	0.25	0.22	-0.19	0.26
gross_tertiary_education_enrollment	0.066	0.063	0.012	-0.46	1	0.52	0.028	-0.39	-0.54	0.11	-0.58
life_expectancy_country	0.022	0.018	-0.05	-0.75	0.52	1	0.36	-0.27	-0.52	0.22	-0.23
tax_revenue_country_country	-0.0092	0.0059	-0.15	-0.037	0.028	0.36	1	-0.049	-0.43	0.026	0.1
total_tax_rate_country	-0.036	-0.14	0.11	0.25	-0.39	-0.27	-0.049	1	0.64	-0.083	0.46
population_country	-0.053	-0.16	0.21	0.22	-0.54	-0.52	-0.43	0.64	1	-0.08	0.47
latitude_country	0.041	-0.14	0.094	-0.19	0.11	0.22	0.026	-0.083	-0.08	1	-0.15
longitude_country	-0.099	-0.17	0.1	0.26	-0.58	-0.23	0.1	0.46	0.47	-0.15	1

Modelling

Modelling

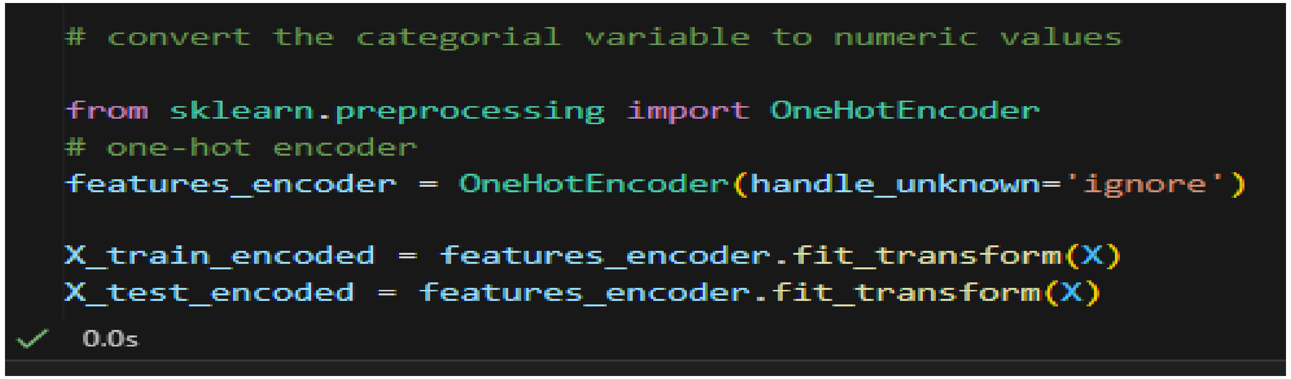
1. Data Splitting
2. Preprocess the Data
3. Model Training
4. Model Fitting
5. Model Evaluation

Logistic Regression



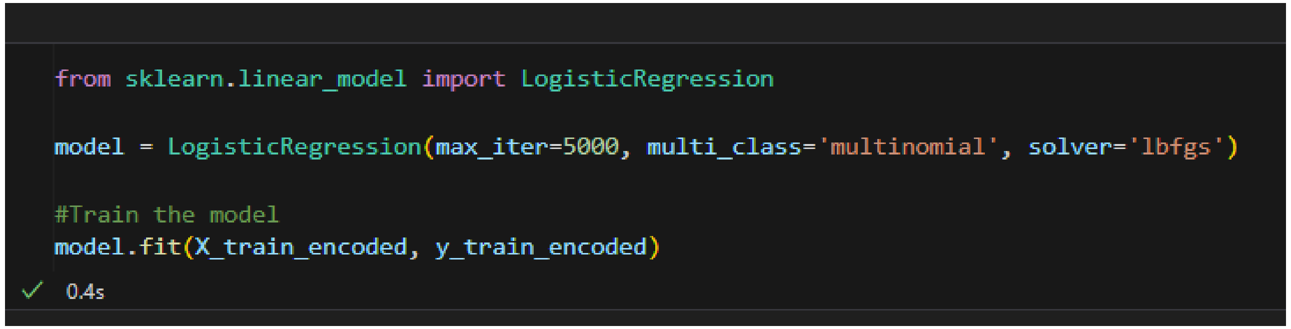
```
from sklearn.model_selection import train_test_split

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
```

✓ 0.7s



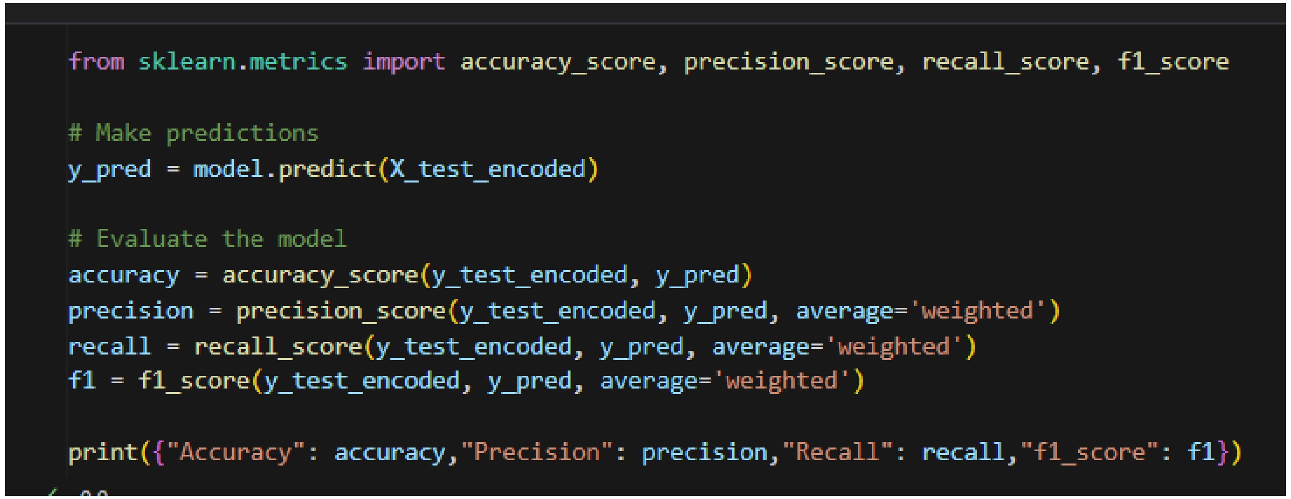
```
# convert the categorial variable to numeric values

from sklearn.preprocessing import OneHotEncoder
# one-hot encoder
features_encoder = OneHotEncoder(handle_unknown='ignore')

X_train_encoded = features_encoder.fit_transform(X)
X_test_encoded = features_encoder.fit_transform(X)
```

✓ 0.0s



```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=5000, multi_class='multinomial', solver='lbfgs')

#Train the model
model.fit(X_train_encoded, y_train_encoded)
```

✓ 0.4s

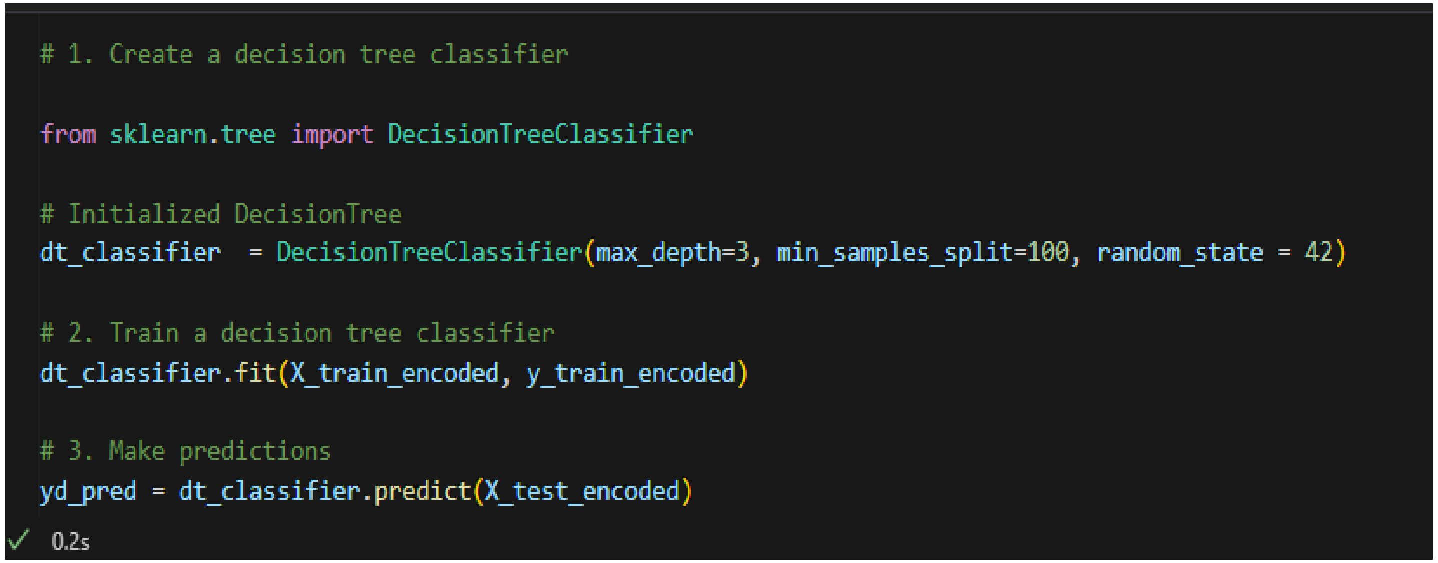


```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions
y_pred = model.predict(X_test_encoded)

# Evaluate the model
accuracy = accuracy_score(y_test_encoded, y_pred)
precision = precision_score(y_test_encoded, y_pred, average='weighted')
recall = recall_score(y_test_encoded, y_pred, average='weighted')
f1 = f1_score(y_test_encoded, y_pred, average='weighted')

print({"Accuracy": accuracy,"Precision": precision,"Recall": recall,"f1_score": f1})
```

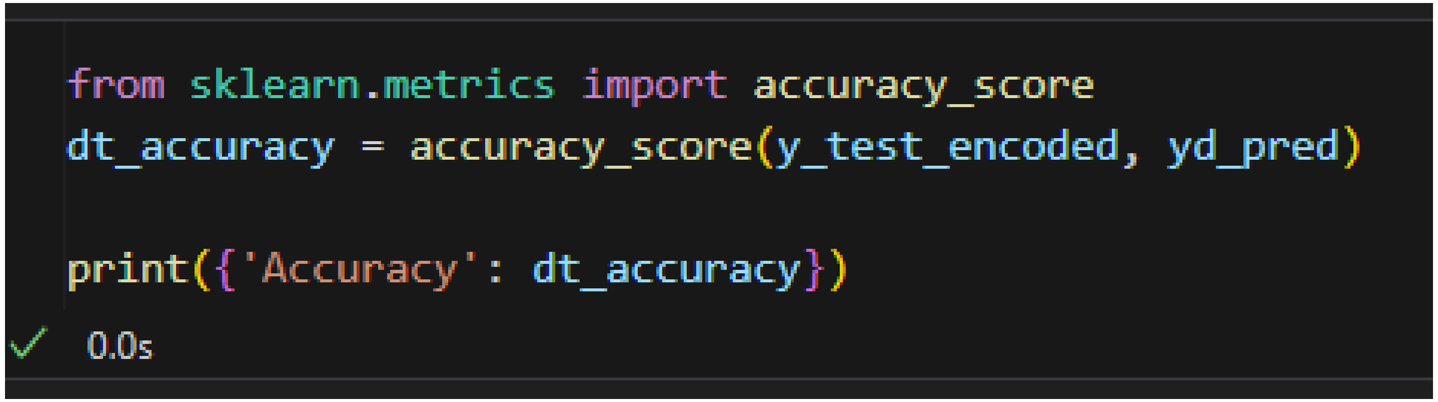
Modelling

1. Data Splitting
2. Preprocess the Data
3. Model Training
4. Model Fitting
5. Model Evaluation

Decision Tree Classifier



```
# 1. Create a decision tree classifier

from sklearn.tree import DecisionTreeClassifier

# Initialized DecisionTree
dt_classifier  = DecisionTreeClassifier(max_depth=3, min_samples_split=100, random_state = 42)

# 2. Train a decision tree classifier
dt_classifier.fit(X_train_encoded, y_train_encoded)

# 3. Make predictions
yd_pred = dt_classifier.predict(X_test_encoded)
✓ 0.2s
```



```
from sklearn.metrics import accuracy_score
dt_accuracy = accuracy_score(y_test_encoded, yd_pred)

print({'Accuracy': dt_accuracy})
✓ 0.0s
```

Findings

Recommendation

Q1	Q2	Q3	Q4
Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum
Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum
Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum	Lorem ipsum et tula lorem ipsum et lorem ipsum

Conclusion

THANK YOU

20TH OCTOBER 2023