Financial Data Analysis: From Collection to ML Models

This document outlines the process of analyzing financial data, starting from data collection, through cleaning and exploratory data analysis (EDA), and finally to the application of machine learning models for prediction.

S by Shubham Talele

Step 01: Data Collection and Initial Overview

The first step involved collecting the financial data. The dataset, named 'financial_data_2.csv', was loaded into a pandas DataFrame. Initial checks were performed to understand its structure and content.

- The `statsmodels` library was installed.
- Essential libraries such as `numpy`, `pandas`, `seaborn`, `matplotlib.pyplot`, and `os` were imported.
- The dataset was loaded using `pd.read_csv('financial_data_2.csv')`.
- Display options were set to show all columns: `pd.set_option('display.max_columns', None)` and `pd.set_option('display.width', None)`.

The shape of the DataFrame was found to be **(500000, 27)**, indicating 500,000 entries and 27 columns. A preview of the first and last 10 rows, as well as specific columns like 'City_Tier' and 'Dependents', was examined. The `df.info()` command revealed the data types and non-null counts for each column, showing that many columns had missing values.

Initial Data Information

Column	Non-Null Count	Dtype
Income	483904	float64
Age	483914	float64
Dependents	484230	float64
Occupation	484226	object
City_Tier	484367	float64
...	...	...
Total Columns	27	dtypes: float64(26), object(1)

The `df.describe()` function provided summary statistics for numerical columns, and `df.dtypes` confirmed the data types.

Step 02: Cleaning Dataset

The dataset cleaning process involved handling missing values, addressing multicollinearity, and managing outliers. The initial check for missing values using `df.isnull().sum()` confirmed their presence, while `df.duplicated().sum()` showed no duplicate rows.

Missing Value Imputation

Missing values were filled based on column type:

- **Categorical-like columns:** 'Occupation' was filled with 'Unknown'.
- **Numerical discrete columns:** 'Age' and 'Dependents' were filled with their respective medians and cast to integer type. 'City_Tier' was filled with its mode and cast to integer.
- **Continuous numeric columns:** All remaining float64 columns were filled with their medians.

After imputation, the dataset shape was **(500000, 27)**, and there were 0 remaining missing values.

Multicollinearity Analysis and Feature Dropping

Multicollinearity was assessed using Variance Inflation Factor (VIF) and correlation matrices. Initially, columns with high multicollinearity such as 'Disposable_Income', 'Healthcare', 'Entertainment', 'Miscellaneous', 'Education', and 'Eating_Out' were dropped. This reduced the dataset shape to **(500000, 22)**.

A VIF calculation was performed on the remaining numeric features. Features with VIF values greater than 10 were identified and dropped. 'Desired_Savings' was identified as a high VIF feature and subsequently dropped, reducing the dataset shape to **(500000, 15)**.

The correlation matrix was computed and visualized using a heatmap. Highly correlated pairs (absolute correlation > 0.8) were identified. 'Potential_Savings_Healthcare' was dropped due to high correlation, resulting in a final dataset shape of **(500000, 14)**.

Outlier Management

Outliers were identified using box plots and quantified by calculating the percentage of outliers in each numerical column using the IQR method. For example:

Income	0.92% outliers
Age	0.00% outliers
Desired_Savings_Percentage	0.46% outliers
Potential_Savings_Groceries	1.89% outliers
Total_Expenses	0.99% outliers

Outliers in all numerical columns were capped using the IQR method to mitigate their impact on model performance. After capping, box plots were re-generated to confirm the removal of extreme outliers.

Step 03: Exploratory Data Analysis (EDA) and Inferential Statistics

Exploratory Data Analysis (EDA) was performed to understand the distributions of features and their relationships with the target variable, 'Total_Expenses'.

Univariate Analysis (Feature Distributions)

Histograms with KDE were generated for all numerical columns to visualize their distributions. For categorical columns, a count plot was created for 'Occupation' to show the frequency of each occupation type.

Bivariate Analysis (Feature vs. Target)

Scatter plots were used to visualize the relationship between each numerical feature and 'Total_Expenses'. Box plots were used to show the relationship between 'Occupation' (categorical) and 'Total_Expenses'.

Correlation Heatmap

A correlation heatmap of all numeric features was generated to visualize the linear relationships between variables. The top 10 most correlated feature pairs were identified:

- Income ↔ Total_Expenses : 0.86
- Total_Expenses ↔ Income : 0.86
- Total_Expenses ↔ Potential_Savings_Groceries : 0.49
- Potential_Savings_Groceries ↔ Total_Expenses : 0.49
- Potential_Savings_Utilities ↔ Total_Expenses : 0.46
- Total_Expenses ↔ Potential_Savings_Utilities : 0.46
- Potential_Savings_Utilities ↔ Income : 0.46
- Income ↔ Potential_Savings_Utilities : 0.46
- Income ↔ Potential_Savings_Groceries : 0.46
- Potential_Savings_Groceries ↔ Income : 0.46

Skewness & Outliers

The skewness of numeric columns was calculated to understand the asymmetry of their distributions. Box plots were also used to visually inspect for outliers after the capping process.

Inferential Statistics

Inferential statistical tests were conducted to draw conclusions about the population based on the sample data.

1 T-Test or ANOVA: Is Total_Expenses significantly different across Occupations?

A one-way ANOVA test was performed to determine if there is a statistically significant difference in 'Total_Expenses' across different 'Occupation' groups. The results were:

- ANOVA F-statistic: 0.25
- P-value: 0.86033

Since the P-value (0.86033) is greater than 0.05, it was concluded that there is **no significant difference** between occupation groups regarding 'Total_Expenses'.

2 Confidence Interval for Mean Desired Savings %

A 95% Confidence Interval for 'Desired_Savings_Percentage' was calculated:

- 95% Confidence Interval for Desired Savings %: 0.15 to 0.15

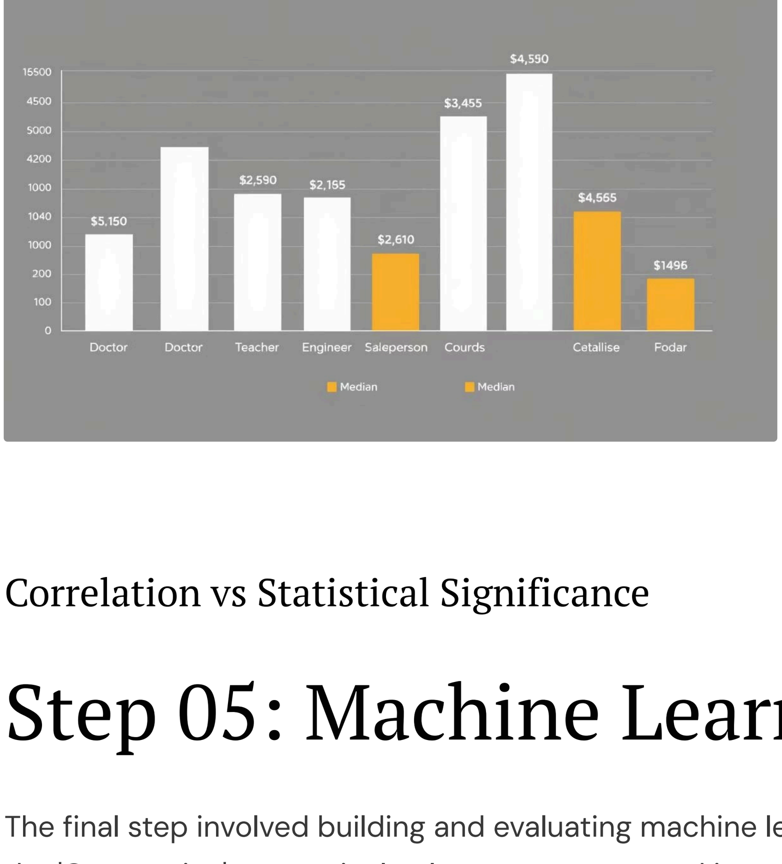
3 Significance of Correlation

Pearson correlation coefficients and p-values were calculated for each numerical feature against 'Total_Expenses'.

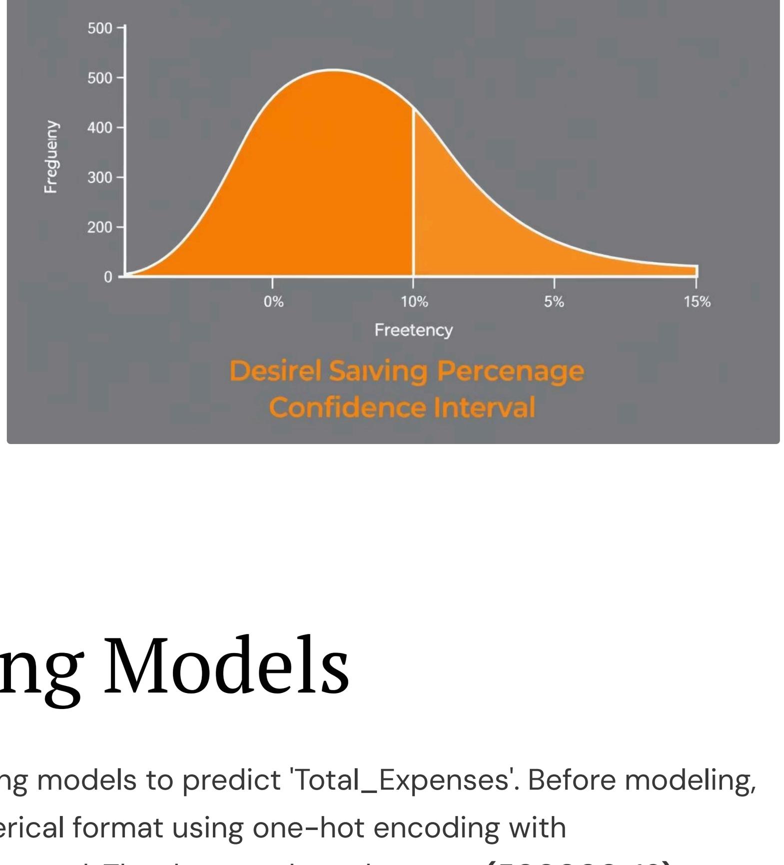
Feature	Correlation (Corr)	P-value (P)
Income	0.86	0.00000
Age	0.00	0.12941
Dependents	-0.00	0.30264
City_Tier	0.00	0.57810
Desired_Savings_Percentage	-0.00	0.40228
Potential_Savings_Groceries	0.49	0.00000
Potential_Savings_Transport	0.45	0.00000
Potential_Savings_Eating_Out	0.37	0.00000
Potential_Savings_Entertainment	0.34	0.00000
Potential_Savings_Utilities	0.46	0.00000
Potential_Savings_Education	0.37	0.00000
Potential_Savings_Miscellaneous	0.37	0.00000

Visualizations of Inferential Statistics

Total_Expenses % by Occupation



95% Confidence Interval for Desired Savings %



Correlation vs Statistical Significance

Step 05: Machine Learning Models

The final step involved building and evaluating machine learning models to predict 'Total_Expenses'. Before modeling, the 'Occupation' categorical column was converted into numerical format using one-hot encoding with pd.get_dummies, and the original 'Occupation' column was dropped. The dataset shape became **(500000, 16)**.

Model Training and Evaluation

The data was split into training and testing sets with a 80/20 ratio using train_test_split (random_state=20). All features were converted to float64 to ensure compatibility with the models.

01-Multi Categorical Regression (OLS Model)

An Ordinary Least Squares (OLS) regression model was fitted using statsmodels.api.OLS. A constant was added to the features for the intercept. The model performance was evaluated using RMSE and R^2 scores.

- Train RMSE: 4249.18
- Test RMSE: 4267.49
- Train R^2: 0.7660
- Test R^2: 0.7658

The OLS model summary provided detailed statistics, including coefficients, p-values, and confidence intervals for each feature.

02-Decision Tree Regressor

A Decision Tree Regressor was trained, and hyperparameters were tuned using GridSearchCV with R^2 as the scoring metric. The best parameters found were {'criterion': 'squared_error', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2}.

- Train R^2: 0.7749
- Test R^2: 0.7653

03-Random Forest Regressor

A Random Forest Regressor was also trained with hyperparameter tuning using GridSearchCV. The best parameters were {'criterion': 'squared_error', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50}.

- Train R^2: 0.7815
- Test R^2: 0.7736

04-Gradient Boosting Regressor

A Gradient Boosting Regressor was trained with random_state=42. Its performance was evaluated on the test set.

- RMSE: 4189.51
- R^2 Score: 0.77
- Train R^2: 0.7752
- Test R^2: 0.7742

The Gradient Boosting Regressor showed slightly better performance in terms of R^2 score on the test set compared to the other models.