python -m venv venv

source venv/bin/activate

git config –list

kedro info

pip install kedro

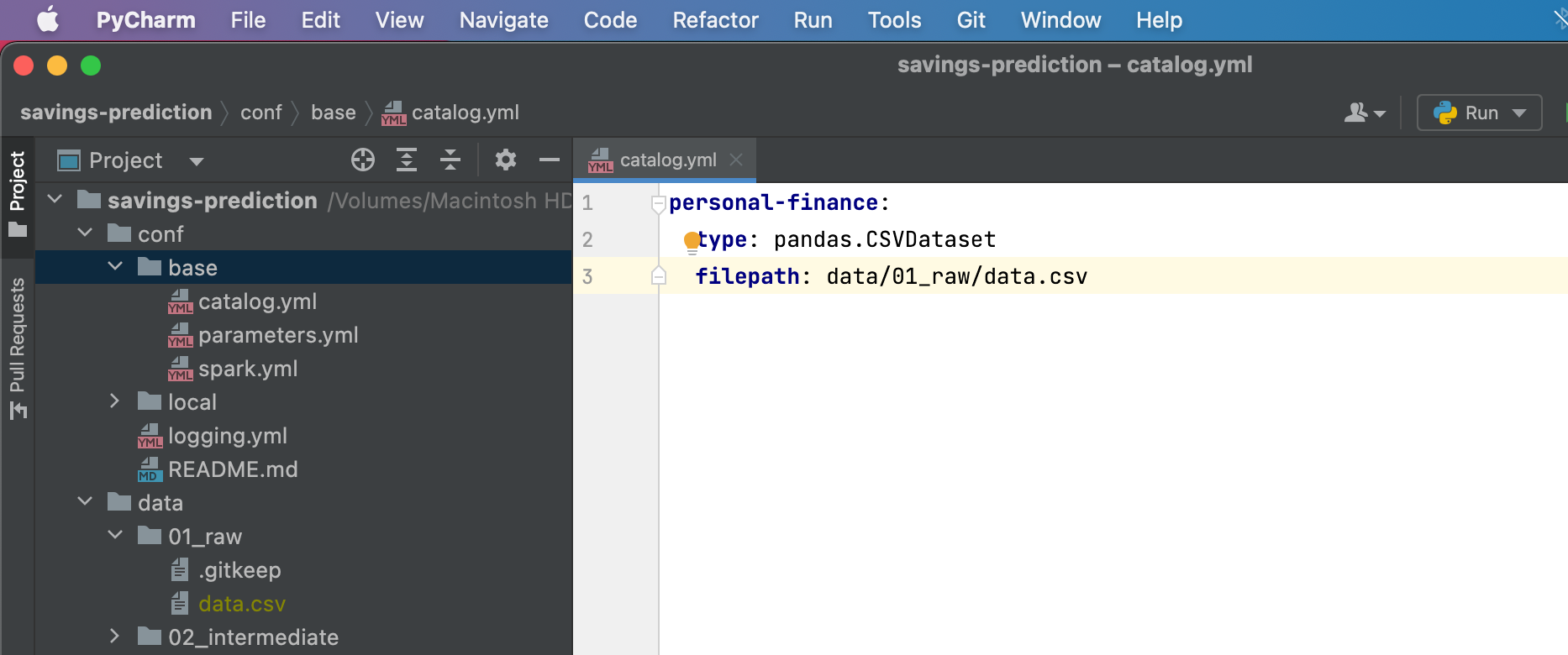
kedro info

kedro new

name – savings prediction

cd savings-prediction

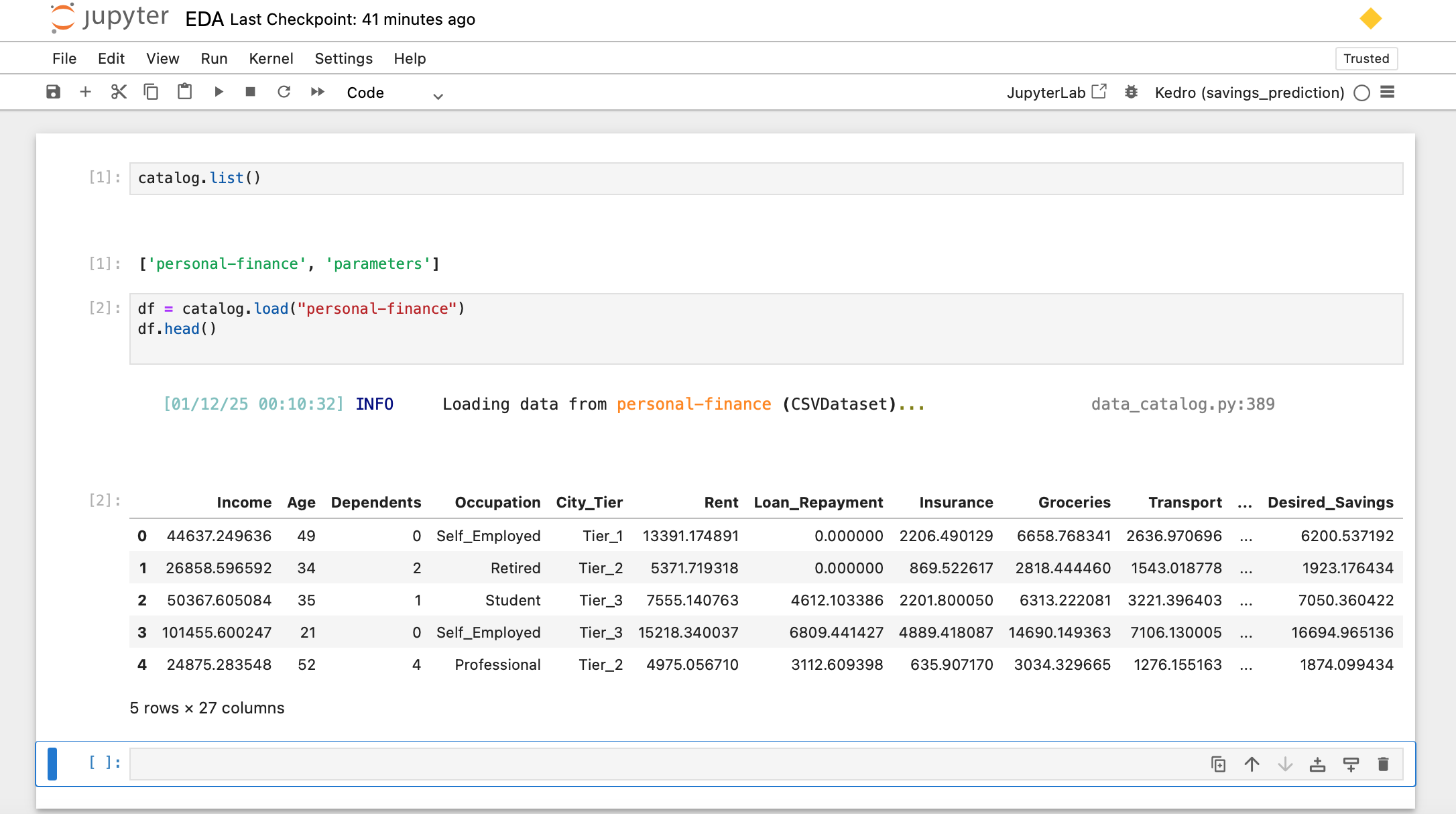
pip install -r requirements.txt



kedro run

kedro jupyter notebook

Ex : (venv) vikrambahadur@Vikrams-MacBook-Pro savings-prediction % kedro jupyter notebook



git init

git add .

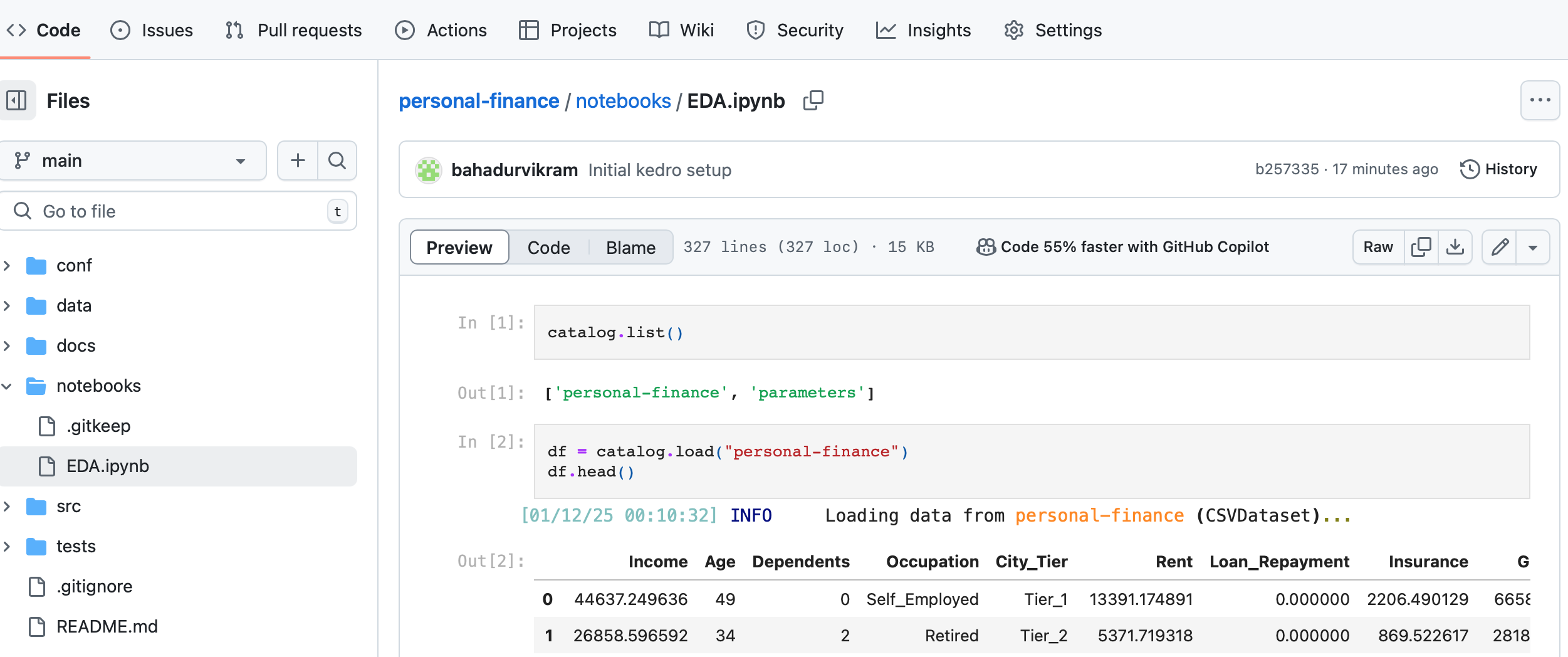
commit -m "Initial kedro setup"

git branch -M main

git remote add origin https://github.com/bahadurvikram/personal-finance.git

git remote -v

git push -u origin main



### ****1. XGBoost (Extreme Gradient Boosting)****

#### **Pros**:

* **Highly Optimized**: Fast implementation with parallel processing and out-of-core computation.
* **Regularization**: Built-in L1 and L2 regularization to prevent overfitting.
* **Handles Missing Data**: Automatically learns the best way to handle missing values.
* **Feature Importance**: Provides multiple ways to calculate feature importance.
* **Wide Adoption**: Large community and extensive documentation.

#### **Cons**:

* **Computational Cost**: Slightly slower compared to LightGBM for very large datasets.
* **Parameter Tuning**: Can be challenging due to the large number of hyperparameters.

#### **Best For**:

* Datasets of all sizes, especially if you require precise feature engineering and flexibility.
* Applications like structured/tabular data with lots of features.

### ****2. LightGBM (Light Gradient Boosting Machine)****

#### **Pros**:

* **Speed**: Faster than XGBoost on large datasets due to histogram-based algorithms and leaf-wise growth.
* **Memory Efficiency**: Lower memory usage compared to XGBoost.
* **Scalability**: Handles large datasets with millions of rows and high feature dimensions efficiently.
* **Categorical Features**: Supports native categorical feature handling without preprocessing.

#### **Cons**:

* **Overfitting Risk**: The leaf-wise splitting strategy can lead to overfitting on small datasets.
* **Requires Clean Data**: Sensitive to outliers and noise.

#### **Best For**:

* Large datasets with many features and high cardinality categorical data.
* When training time is a constraint.

### ****3. CatBoost (Categorical Boosting)****

#### **Pros**:

* **Handles Categorical Features**: Excellent at handling categorical data natively, without the need for one-hot encoding.
* **Less Hyperparameter Tuning**: Works well with minimal tuning.
* **Built-in Handling of Overfitting**: Has features like ordered boosting to reduce overfitting.
* **Ease of Use**: Simple to implement and often provides strong out-of-the-box performance.

#### **Cons**:

* **Speed**: Slightly slower than LightGBM for very large datasets.
* **Limited Community Support**: Smaller community compared to XGBoost and LightGBM.

#### **Best For**:

* Datasets with a significant number of categorical variables.
* Projects where ease of use and reduced preprocessing are priorities.
* **Comparison Table**

| **Feature** | **XGBoost** | **LightGBM** | **CatBoost** |
| --- | --- | --- | --- |
| **Speed** | Moderate | Fastest | Fast |
| **Memory Usage** | Moderate | Low | Moderate |
| **Categorical Data Handling** | Requires Encoding | Native Support | Native Support |
| **Overfitting Prevention** | Regularization | Leaf-Wise Splitting | Ordered Boosting |
| **Ease of Use** | Moderate | Moderate | High |
| **Community & Documentation** | Large | Growing | Smaller |
| **Best For** | Versatile Datasets | Large Datasets | Categorical Data |

**2. LightGBM (Light Gradient Boosting Machine) -** Sensitive to outliers and noise.

 Large datasets with many features and high cardinality categorical data.

27

Occupation

City\_Tier

Age (num) -

Dependents (num) -

Occupation (cat)

City\_Tier (cat)

Rent (fixed) - remove

Loan\_Repayment (fixed) - remove

Insurance (fixed) - remove

Groceries (ve - y)

Transport (ve - y)

Eating\_Out (ve - y)

Entertainment(ve - y)

Utilities(ve - y)

Healthcare(ve - y)

Education(ve - y)

Miscellaneous(ve - y)

Desired\_Savings\_Percentage

Desired\_Savings (num)

Disposable\_Income (num)

Potential\_Savings\_Groceries (T)

Potential\_Savings\_Transport (T)

Potential\_Savings\_Eating\_Out (T)

Potential\_Savings\_Entertainment (T)

Potential\_Savings\_Utilities (T)

Potential\_Savings\_Healthcare (T)

Potential\_Savings\_Education (T)

Potential\_Savings\_Miscellaneous (T)

Income (num)

kedro ipython

%load\_ext kedro.ipython

%reload\_ext kedro.ipython

kedro pipeline create dataprocessing

kedro registry list

kedro run -–pipeline dataprocessing

kedro pipeline create data\_science