**ISM540 TEAM PROJECT REPORT**

**SHOP CUSTOMER DATA**

**DATASET DESCRIPTION**

Shop Customer Data is a thorough examination of a creative shop's ideal clients. It aids in a company's better understanding of its clients. Membership cards provide information about Customers to a shop owner.

Dataset has 8 columns and 2000 records:

* Customer ID
* Gender
* Age
* Annual Income
* Spending Score: The shop assigns this score based on the customer's spending habits and behavior.
* Profession
* Years of Work Experience
* Family Size
* Source:<https://www.kaggle.com/datasets/datascientistanna/customers-dataset?resource=download>

**Dataset Description**

The spending score is the class variable, and the predictor variables are gender, age, annual income, profession, work experience, and family size. This dataset could be beneficial for businesses to understand their customers better and make informed decisions about their marketing strategies, product offerings, and overall business operations.

By analyzing this dataset, businesses can identify patterns and insights that can help them segment their customers based on various characteristics such as income, age, and spending behavior. This segmentation can help businesses personalize their marketing efforts and tailor their product offerings to specific customer groups, which can ultimately lead to increased profits.

In summary, the Shop Customer Data dataset can be a valuable resource for businesses to gain insights into their customers' behavior and preferences, enabling them to make data-driven decisions to optimize their operations and maximize profits.

**OPERATOR DESCRIPTION**

* **Split Data:**

This operator produces the desired number of subsets of the given ExampleSet. The ExampleSet is partitioned into subsets according to the specified relative sizes.

The Split Data operator takes an ExampleSet as its input and delivers the subsets of that ExampleSet through its output ports. The number of subsets (or partitions) and the relative size of each partition are specified through the partitions parameter. The sum of the ratio of all partitions should be 1. The sampling type parameter decides how the examples should be shuffled in the resultant partitions. For more information about this operator please study the parameters section of this description. This operator is different from other sampling and filtering operators in the sense that it is capable of delivering multiple partitions of the given ExampleSet.

* **Select Attributed**

This Operator selects a subset of Attributes of an ExampleSet and removes the other Attributes. The Operator provides different filter types to make Attribute selection easy. Possibilities are for example: Direct selection of Attributes. Selection by a regular expression or selecting only Attributes without missing values. See parameter *attribute filter type* for a detailed description of the different filter types.

The *type* parameter can be used to decide whether to include or exclude the selected Attributes. Special Attributes (Attributes with Roles, like id, label, weight) are by default ignored in the selection. They will always remain in the resulting output ExampleSet. The parameter *also apply to special attributes* changes this.

Only the selected Attributes are delivered to the output port. The rest are removed from the ExampleSet.

* **Set Role**

This Operator is used to change the role of one or more Attributes. The role of an Attribute describes how other Operators handle this Attribute. The default role is regular, other roles are classified as special. The different types of roles are explained below in the parameter section.

An ExampleSet can have many special Attributes and you can assign special Attributes multiple times. This comes in handy, for example, if you want to feed the Attributes into a learner that accepts multiple labels. However, please note that some operators expect the special roles to be unique and they might not know how to handle duplicate special roles.

* **Nominal to Numerical**

This operator changes the type of selected non-numeric attributes to a numeric type. It also maps all values of these attributes to numeric values.

The Nominal to Numerical operator is used for changing the type of non-numeric attributes to a numeric type. This operator not only changes the type of selected attributes but it also maps all values of these attributes to numeric values. Binary attribute values are mapped to 0 and 1. Numeric attributes of input the ExampleSet remain unchanged. This operator provides three modes for conversion from nominal to numeric. This mode is selected by the coding type parameter. Explanation of these coding types is given in the parameters and they are also explained in the example process.

* **Linear Regression**

This operator calculates a linear regression model from the input ExampleSet.

Regression is a technique used for numerical prediction. Regression is a statistical measure that attempts to determine the strength of the relationship between one dependent variable ( i.e. the label attribute) and a series of other changing variables known as independent variables (regular attributes). Just like Classification is used for predicting categorical labels, Regression is used for predicting a continuous value. For example, we may wish to predict the salary of university graduates with 5 years of work experience, or the potential sales of a new product given its price. Regression is often used to determine how much specific factors such as the price of a commodity, interest rates, particular industries or sectors influence the price movement of an asset.

Linear regression attempts to model the relationship between a scalar variable and one or more explanatory variables by fitting a linear equation to observed data. For example, one might want to relate the weights of individuals to their heights using a linear regression model.

This operator calculates a linear regression model. It uses the Akaike criterion for model selection. The Akaike information criterion is a measure of the relative goodness of a fit of a statistical model. It is grounded in the concept of information entropy, in effect offering a relative measure of the information lost when a given model is used to describe reality. It can be said to describe the tradeoff between bias and variance in model construction, or loosely speaking between accuracy and complexity of the model.

* **Apply Model**

This Operator applies a model on an ExampleSet.

A model is first trained on an ExampleSet by another Operator, which is often a learning algorithm. Afterwards, this model can be applied on another ExampleSet. Usually, the goal is to get a prediction on unseen data or to transform data by applying a preprocessing model.

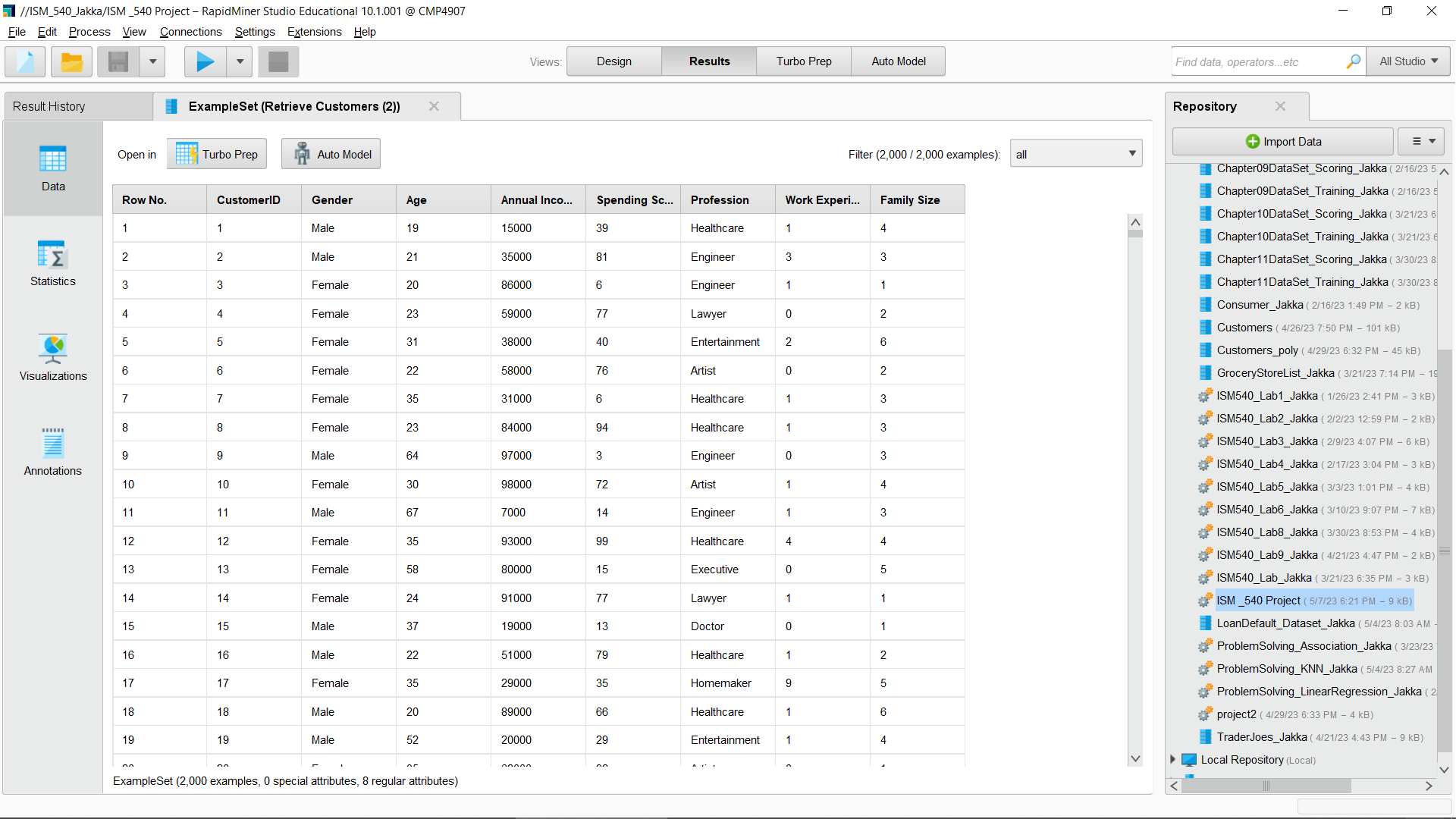
The ExampleSet upon which the model is applied, has to be compatible with the Attributes of the model. This means, that the ExampleSet has the same number, order, type and role of Attributes as the ExampleSet used to generate the model.

* **Aggregate**

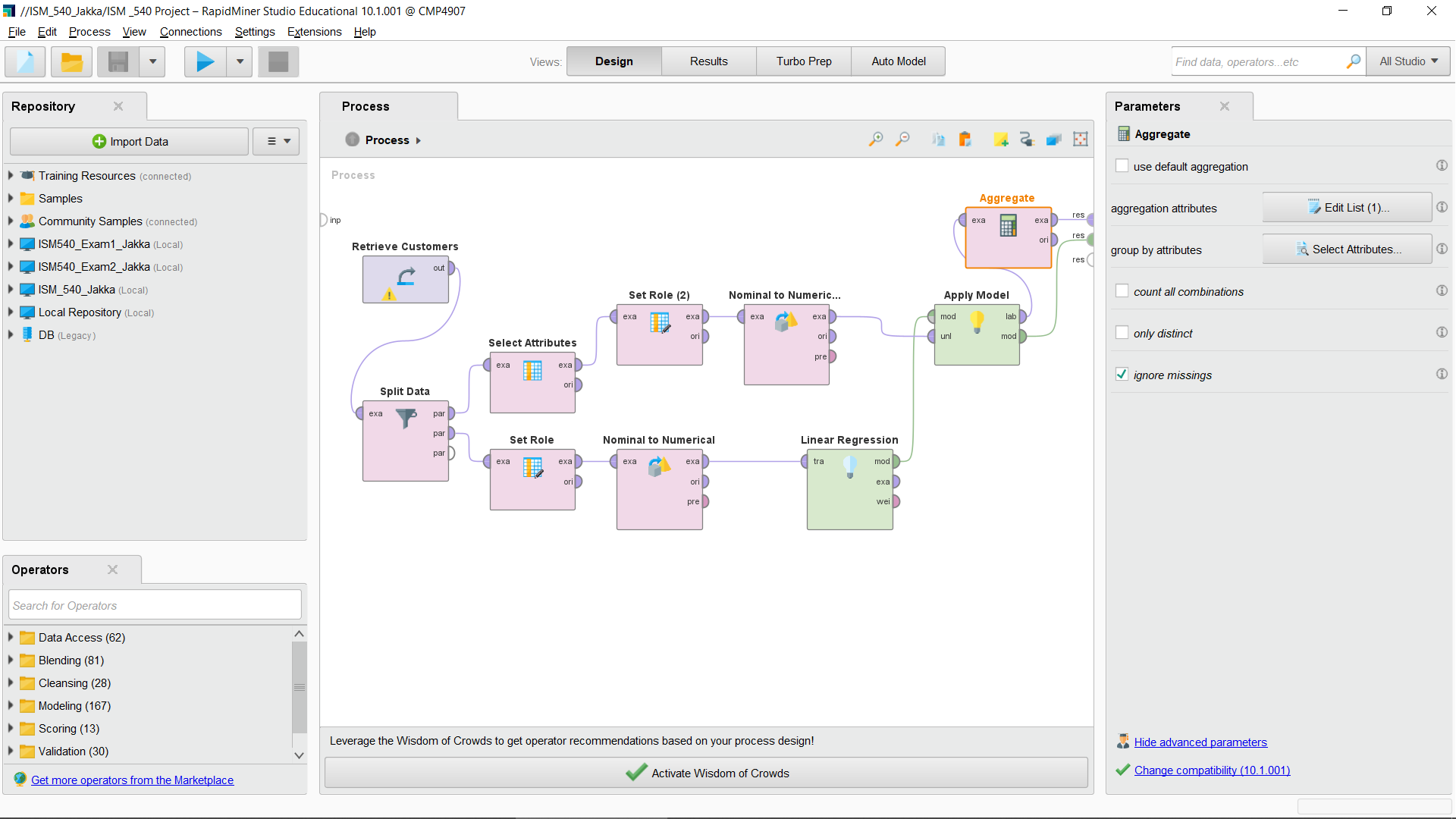
This operator performs the aggregation functions known from SQL. This operator provides a lot of functionalities in the same format as provided by the SQL aggregation functions. SQL aggregation functions and GROUP BY and HAVING clauses can be imitated using this operator.

The Aggregate operator creates a new ExampleSet from the input ExampleSet showing the results of the selected aggregation functions. Many aggregation functions are supported including SUM, COUNT, MIN, MAX, AVERAGE and many other similar functions known from SQL. The functionality of the GROUP BY clause of SQL can be imitated by using the group by attributes parameter. You need to have a basic understanding of the GROUP BY clause of SQL for understanding the use of this parameter because it works exactly the same way. If you want to imitate the known HAVING clause from SQL, you can do that by applying the Filter Examples operator after the Aggregation operator. This operator imitates aggregation functions of SQL. It focuses on obtaining summary information, such as averages and counts etc. It can group examples in an ExampleSet into smaller sets and apply aggregation functions on those sets. Please study the attached Example Process for better understanding of this operator.

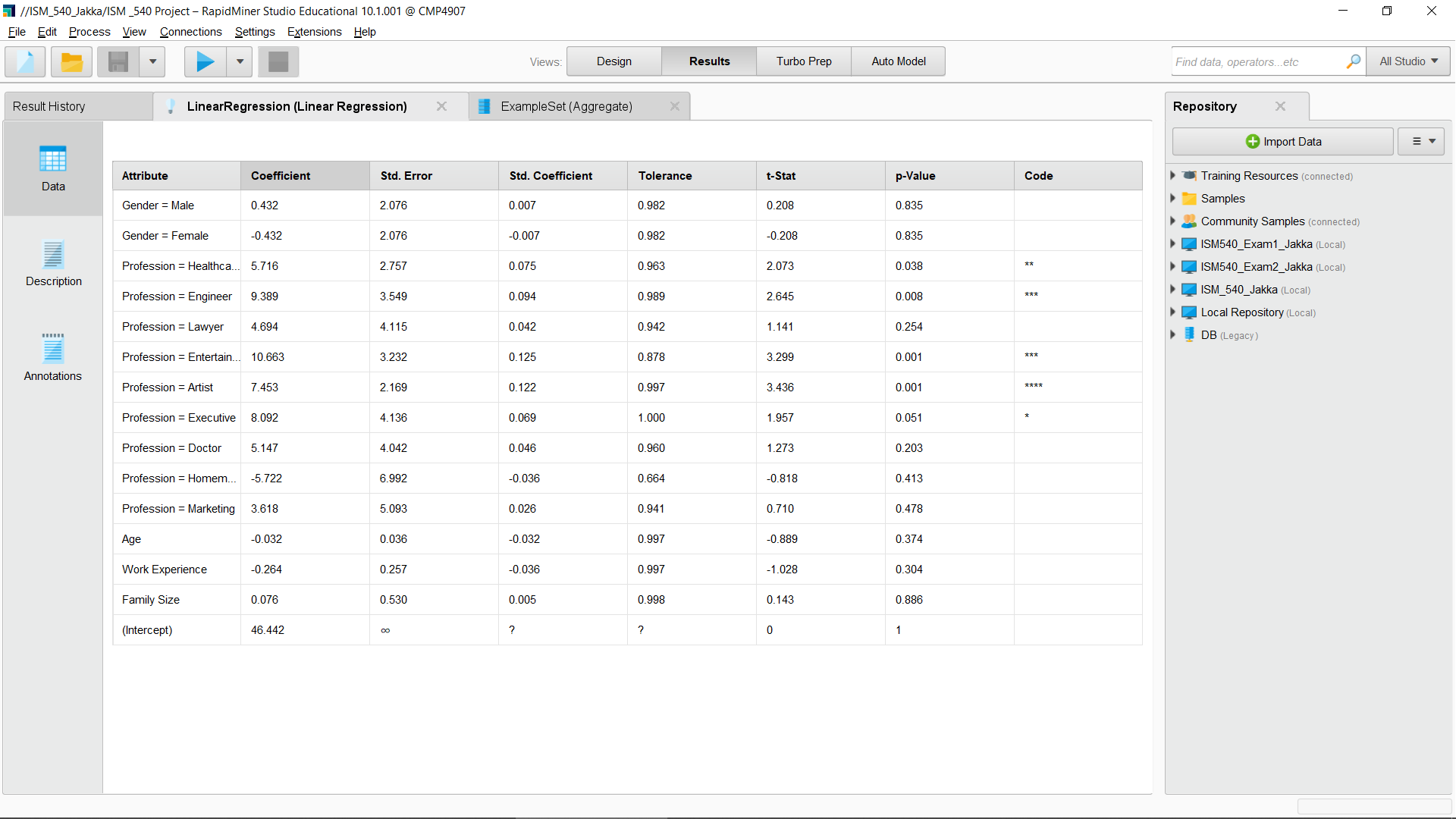
**INPUT**

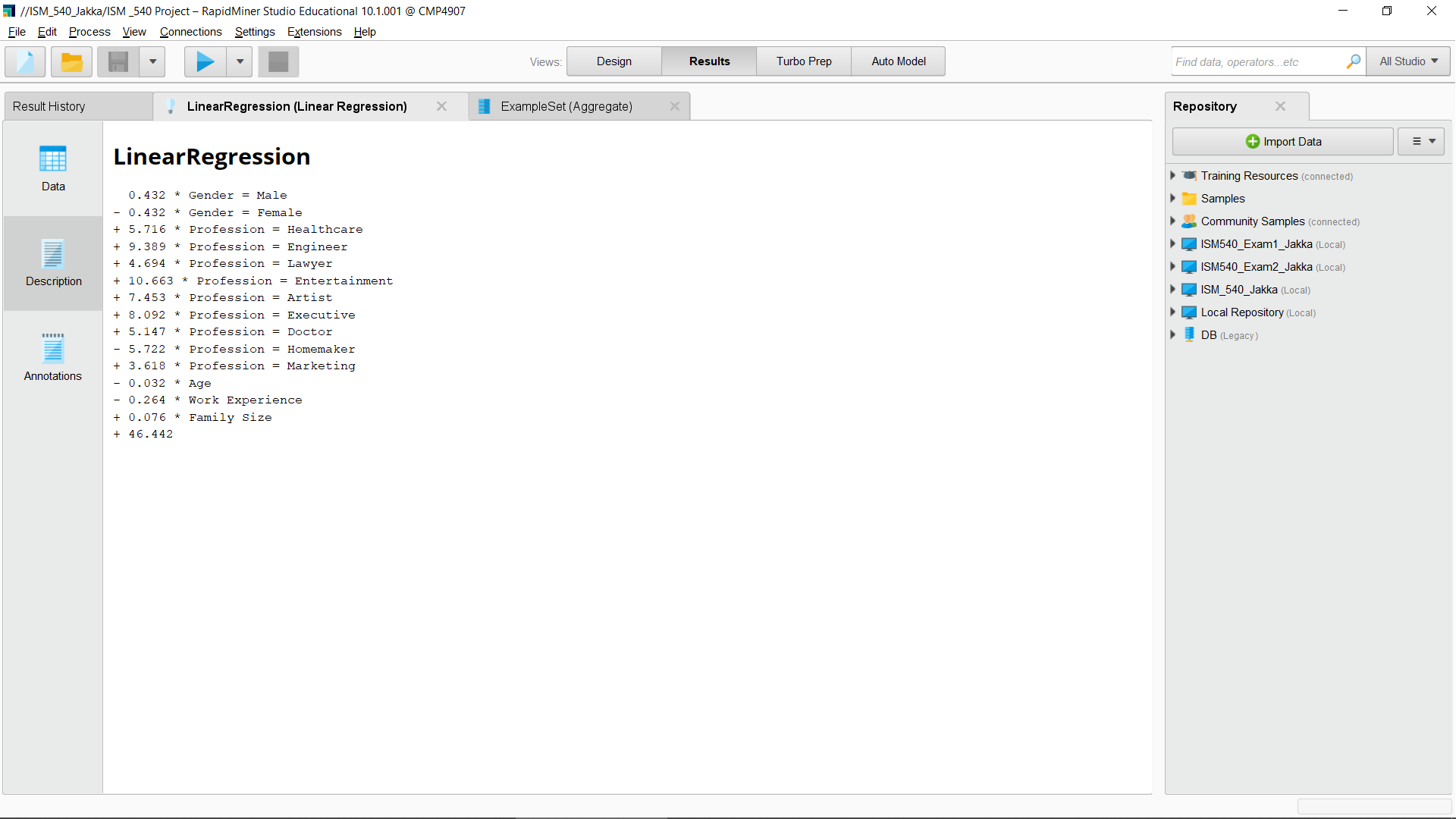


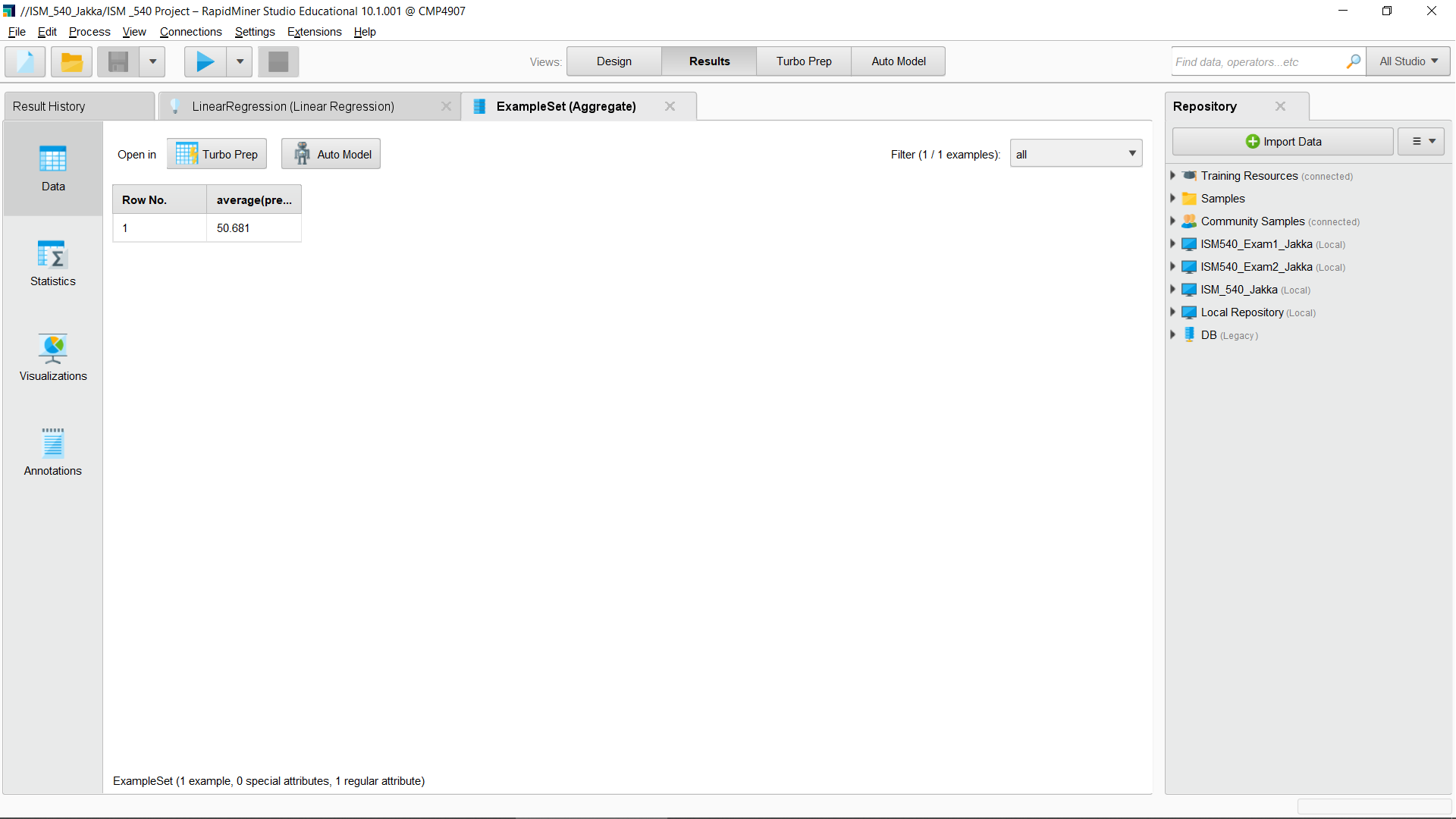
**MODEL**



**OUTPUT**







The above results represent a Linear Regression model that aims to predict a person's income based on their gender, profession, age, work experience, and family size.

The model predicts that being male would increase a person's income by 0.432 units compared to being female. Similarly, the model predicts that having a profession in Healthcare, Engineering, Law, Entertainment, Art, Executive, or Medicine would increase a person's income by 5.716, 9.389, 4.694, 10.663, 7.453, 8.092, or 5.147 units, respectively. Conversely, having a profession in Homemaking would decrease a person's income by 5.722 units compared to the base category.

The model also takes into account the person's age, work experience, and family size as predictors of income. The coefficients for these variables suggest that for every unit increase in age, the predicted income decreases by 0.032 units, for every unit increase in work experience, the predicted income decreases by 0.264 units, and for every unit increase in family size, the predicted income increases by 0.076 units.

Finally, the model has an intercept of 46.442, which is the predicted income for a person with all predictor variables set to zero.

**DIFFICULTIES FACED**

When we first chose the dataset our choice of algorithm was logistic regression while we were trying to implement the algorithm we realized that our attributes had continuous numerical values and logistic regression is not well suited for this kind of data , so we have switched to the linear regression algorithm. While, implementing the model we went through some problems like missing values which we have cleaned.

**WORKING PLAN**

* Dataset – Vaishnavi
* Project Implementation – Surya, Nikhitha
* Presentations – Kiran
* Documentation – Vinuthna