**Name: Ashwin Kalekar**

**Roll No: BEAD21143**

**Subject: Computer Laboratory-IV**

**Class : BE**

**Branch: AI & DS**

**ASSIGNMENT No: 01**

**Title:-**

Implement a Linear Regression Model to predict house prices for regions in the USA using the provided dataset.



Develop a model to estimate house prices based on relevant features using Linear Regression.

* Apply Linear Regression in a real-world scenario.
* Understand the implementation of a regression model for house price prediction.



* Python (3.x recommended)
* Jupyter Notebook or any Python IDE



A machine with sufficient RAM and processing power for model training (8GB RAM recommended)



* Basic understanding of Python programming
* Familiarity with the concepts of supervised learning

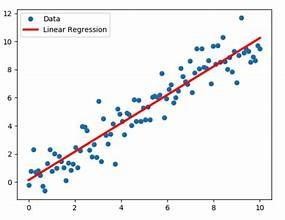


https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price- Prediction/blob/master/USA\_Housing.csv

LINEAR REGRESSION MODEL -

•

NumPy



* pandas
* scikit-learn
* Matplotlib

•

Seaborn



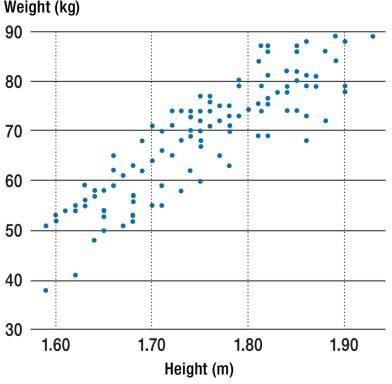
When we see a relationship in a scatterplot, we can use a line to summarize the relationship in the data. We can also use that line to make predictions in the data. This process is called .Linear Regression is a supervised learning algorithm used for predicting a continuous outcome, typically represented by the target variable. In the context of this assignment, we aim to predict house prices based on various features such as area income, house age, number of rooms, and others.



Linear regression is used to study the linear relationship between a dependent variable Y (blood pressure) and one or more independent variables X (age, weight, sex).

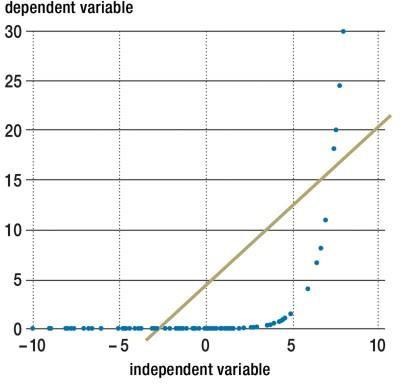
The dependent variable Y must be continuous, while the independent variables may be either continuous (age), binary (sex), or categorical (social status). The initial judgment of a possible relationship between two continuous variables should always be made on the basis of a scatter

plot (scatter graph). This type of plot will show whether the relationship is linear (figure 1) or nonlinear (figure 2).



|  |
| --- |
| Figure 1 |

A scatter plot showing a linear relationship



|  |
| --- |
| Figure 2  A scatter plot showing an exponential relationship. In this case, it would not be appropriate to compute a coefficient of |



The formula for a simple linear regression is:



•  is the predicted value of the dependent variable ( ) for any given value of the

independent variable ( ).

*  is the  , the predicted value of  when the  is 0.
*  is the regression coefficient – how much we expect  to change as  increases.
*  is the independent variable ( the variable we expect is influencing ).
*  is the  of the estimate, or how much variation there is in our estimate of the regression coefficient.

Linear regression finds the line of best fit line through your data by searching for the regression coefficient (B1) that minimizes the total error (e) of the model.

While you can perform a linear regression by hand, this is a tedious process, so most people use statistical programs to help them quickly analyze the data.



Training a regression model involves teaching the model to predict continuous values based on input features. Here's a brief explanation of the process, along with some images to illustrate key concepts.

Regression Model Training:

1. Data Collection: Gather a dataset with input features (independent variables) and corresponding target values (dependent variable).
2. Data Splitting: Split the dataset into training and testing sets. The training set is used to train the model, and the testing set is used to evaluate its performance.
3. Model Selection: Choose a regression model architecture. Common choices include linear regression, decision trees, or more complex models like neural networks.
4. Feature Scaling: Normalize or standardize the input features to ensure that they are on a similar scale. This helps the model converge faster during training.
5. Model Training: Feed the training data into the chosen model and adjust the model's parameters to minimize the difference between predicted and actual target values.
6. Loss Function: Use a loss function to measure the difference between predicted and actual values. The goal is to minimize this loss during training.
7. Gradient Descent: Use optimization algorithms like gradient descent to iteratively update the model parameters and reduce the loss.
8. Model Evaluation: Evaluate the trained model on the testing set to assess its performance on unseen data.
9. Prediction: Once satisfied with the model's performance, use it to make predictions on new, unseen data.

10 Model Deployment: If the model performs well, deploy it to production for making real-world predictions.



The performance of the model will be assessed using various evaluation metrics, such as Mean Squared Error (MSE) and R-squared. These metrics provide insights into how well the model

generalizes to unseen data.

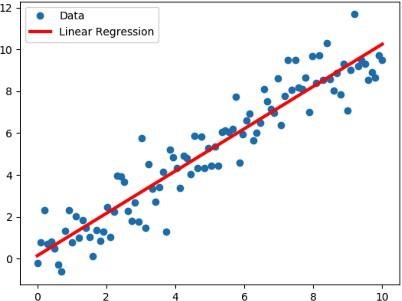


Figure: Linear Regression Model



You ll start with the simplest case, which is simple linear regression. There are five basic steps when you re implementing linear regression:

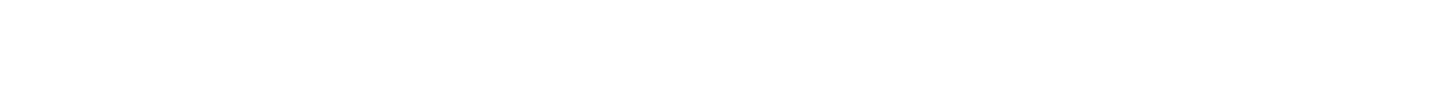
1. Import the packages and classes that you need.
2. Provide data to work with, and eventually do appropriate transformations.
3. Create a regression model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions.

These steps are more or less general for most of the regression approaches and implementations. Throughout the rest of the tutorial, you’ll learn how to do these steps for several different scenarios.



The first step is to import the package numpy and the class LinearRegression from sklearn.linear\_model:

>>> from sklearn.linear\_model import LinearRegression



>>>

import

numpy

as

np



Now, you have all the functionalities that you need to implement linear regression.

The fundamental data type of NumPy is the array type called numpy.ndarray. The rest of this tutorial uses the term  to refer to instances of the type numpy.ndarray.

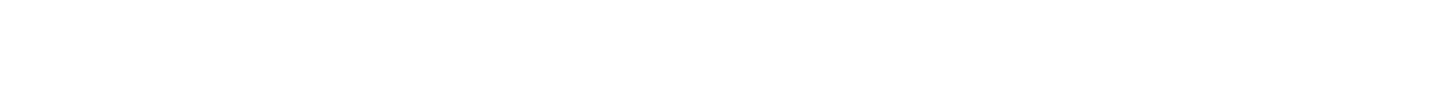
You’ll use the class sklearn.linear\_model.LinearRegression to perform linear and polynomial regression and make predictions accordingly.



The second step is defining data to work with. The inputs (regressors, 𝑥 ) and output (response,

) should be arrays or similar objects. This is the simplest way of providing data for regression:

>>> y = np.array([5, 20, 14, 32, 22, 38])



>>>

x

=

np

.

array([

5

,

15

,

25

,

35

,

45

,

55

])

.

reshape((

-

1

,

1

))



Now, you have two arrays: the input, x, and the output, y. You should call .reshape() on x because this array must

be  , or more precisely, it must have

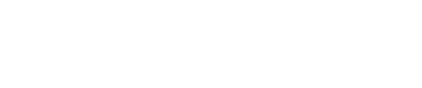
and



. That’s exactly what the argument (-1, 1) of .reshape() specifies.

This is how x and y look now:

As you can see, x has two dimensions, and x.shape is (6, 1), while y has a single dimension, and y.shape is (6,).



array([

5

,

,

20

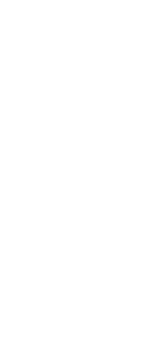
, 32,

14

,

22

38])



array([[

,

5]

[15]

,

[25]

,

[35]

,





The next step is to create a linear regression model and fit it using the existing data.

Create an instance of the class LinearRegression, which will represent the regression model

and output, x and y, as the arguments. In other words, .fit() . It returns self, which is the variable model itself. That’s why you can replace the last two statements with this one:

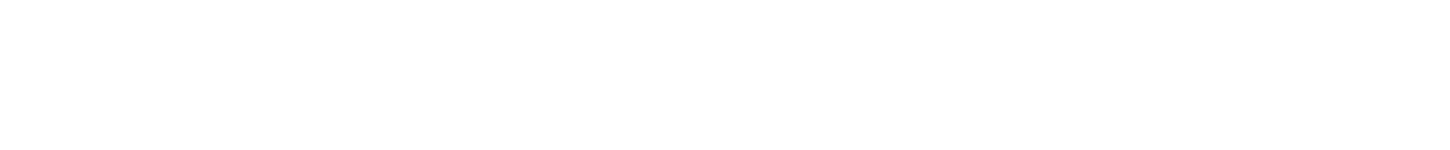
>>> model = LinearRegression().fit(x, y)



This statement does the same thing as the previous two. It’s just shorter.



Once you have your model fitted, you can get the results to check whether the model works satisfactorily and to interpret it. You can obtain the coefficient of determination, 𝑅 ², with .score() called on model:



>>>

r\_sq

=

model

.

score(x,

y)

>>>

print

(

f

"coefficient of determination:

{

r\_sq

}

"

)



When you’re applying .score(), the arguments are also the predictor x and response y, and the

return value is 𝑅 ².

The attributes of model are .intercept\_, which represents the coefficient 𝑏 ₀, and .coef\_, which

represents 𝑏 ₁:

|  |
| --- |
|  |
| >>> print(f"intercept: {model.intercept\_}") intercept: 5.633333333333329  >>> print(f"slope: {model.coef\_}") |

slope: [0.54]

The code above illustrates how to get 𝑏 o and 𝑏 ₁. You can notice that .intercept\_ is a scalar, while .coef\_ is an array.

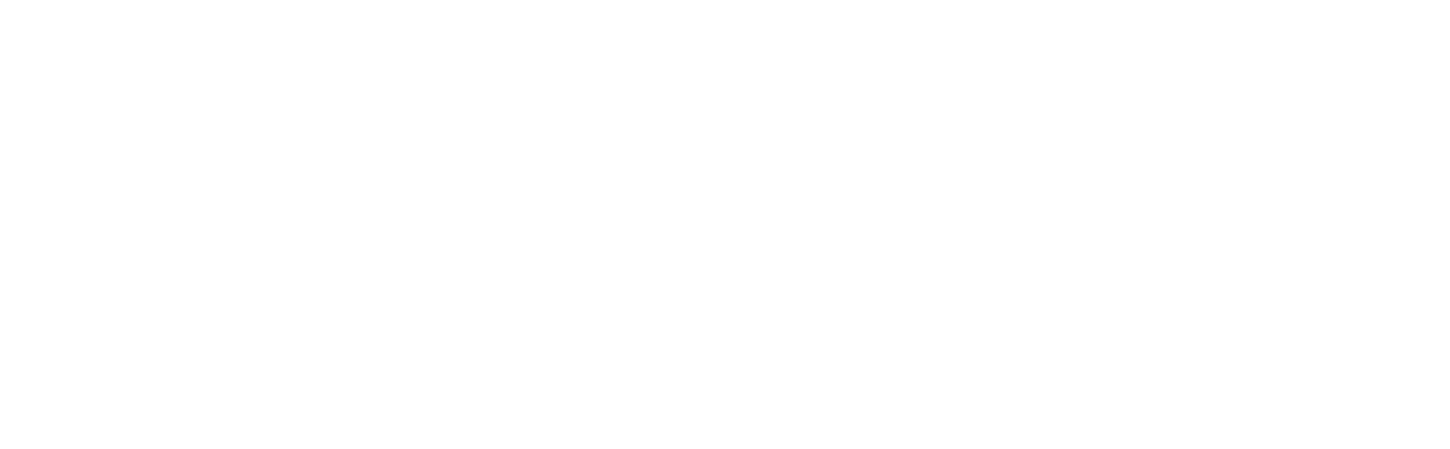
The value of o is approximately 5.63. This illustrates that your model predicts the response 5.63 when is zero. The value ₁ = 0.54 means that the predicted response rises by 0.54 when

𝑥 is increased by one.

You’ll notice that you can provide y as a two-dimensional array as well. In this case, you’ll get a similar result. This is how it might look:

|  |
| --- |
| predicted response:  [ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333] |

When applying .predict(), you pass the regressor as the argument and get the corresponding predicted response. This is a nearly identical way to predict the response:



>>>

y\_pred

=

model

.

intercept\_

+

model

.

coef\_

\*

x

>>>

print

(

f

"predicted response:

\

n

{

y\_pred

}

"

)

predicted

response:



In this case, you multiply each element of x with model.coef\_ and add model.intercept\_ to the product.

The output here differs from the previous example only in dimensions. The predicted response is now a twodimensional array, while in the previous case, it had one dimension.

If you reduce the number of dimensions of x to one, then these two approaches will yield the same result. You can do this by replacing x with x.reshape(-1), x.flatten(), or x.ravel() when multiplying it with model.coef\_.



estate pricing strategies

1. Assisting clients in making informed decisions
2. Market analysis for regions in the USA
3. Predictive tool for estimating house prices

**Reference : -**

process involves exploring the data, data collection, model fitting, and interpretation of regression results. Practical skills in using statistical software for regression analysis are likely to be developed, enhancing students ability to analyze relationships between variables and make predictions based on linear model.

References – https://en.wikipedia.org/wiki/Linear\_regression

https://[www.khanacademy.org/math/statistics-probability/describing-relationships-](http://www.khanacademy.org/math/statistics-probability/describing-relationships-)  quantitativedata/introduction-to-trend-lines/a/linear-regression-review

https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price- Prediction/blob/master/linear-regression-model.jpg

**Conclusion:** Thus Implemented a Linear Regression Model to predict house prices for regions in the USA using the provided dataset.

**Name: Ashwin Kalekar**

**Roll No: BEAD21143**

**Subject: Computer Laboratory-IV**

**Class : BE**

**Branch: AI & DS**

**ASSIGNMENT No: 02**

**Title:-** Build a Multiclass classifier using the CNN model. Use MNIST or any other suitable dataset. a. Perform Data Pre-processing

1. Define Model and perform training
2. Evaluate Results using confusion matrix.



Building a multiclass classifier using a Convolutional Neural Network (CNN) using MNIST or any other suitable dataset. It involves several steps, including data pre-processing, defining the model architecture, training the model, and evaluating its performance using a confusion matrix.

Implement the technique of Convolution neural network (CNN)



* + Python (3.x recommended)
  + TensorFlow (Deep learning framework for building CNNs)
  + Jupyter Notebook, any Python IDE, or Google Colab (for running Python code)



* + A machine with at least 8GB of RAM is recommended for model training.
  + A multi-core CPU is suitable, and for faster training, a GPU (Graphics Processing Unit) is highly recommended.



* + Basic understanding of Python programming
  + Familiarity with the concepts of Neural Networks, especially Convolutional Neural Networks (CNNs)



https://github.com/AmritK10/MNIST-CNN

•

Numpy

-

for

linear

algebra.



* Pandas - for data analysis.
* Matplotlib - for data visualization. • Tensorflow - for neural networks.



ANN or Artificial Neural Network is a multi-layer fully-connected neural net that consists of many layers, including an input layer, multiple hidden layers, and an output layer. This is a very popular deep learning algorithm used in various classification tasks like audio and words Similarly, we have Convolutional Neural Networks(CNNs) for image classification.

 is basically a model known to be and in recent times it has gained a lot of popularity because of its usefulness. CNN uses multilayer perceptrons to do computational works. CNN uses relatively little pre-processing compared to other image classification algorithms. This means the network learns through filters that in traditional algorithms were hand-engineered. So, for the image processing tasks CNNs are the best- suited option.



Applying a Convolutional Neural Network (CNN) on the MNIST dataset is a popular way to learn about and demonstrate the capabilities of CNNs for image classification tasks. The

MNIST dataset consists of 28×28 grayscale images of hand-written digits (0-9), with a training set of 60,000 examples and a test set of 10,000 examples.



1. Load and preprocess the data: The MNIST dataset can be loaded using the Keras library, and the images can be normalized to have pixel values between 0 and 1.
2. Define the model architecture: The CNN can be constructed using the Keras Sequential API, which allows for easy building of sequential models layer-by-layer. The architecture should typically include convolutional layers, pooling layers, and fully-connected layers.
3. Compile the model: The model needs to be compiled with a loss function, an optimizer, and a metric for evaluation.
4. Train the model: The model can be trained on the training set using the Keras fit() function. It is important to monitor the training accuracy and loss to ensure the model is converging properly.
5. Evaluate the model: The trained model can be evaluated on the test set using the Keras evaluate() function. The evaluation metric typically used for classification tasks is accuracy.



mnist dataset is a dataset of handwritten images as shown below in the image



We can get 99.06% accuracy by using CNN(Convolutional Neural Network) with a functional model. The reason for using a functional model is to maintain easiness while connecting the layers.

1.



* Import necessary libraries, including TensorFlow and Keras.

2.



* Load the MNIST dataset, which consists of 28x28 grayscale images of handwritten digits (0 through

9).

* Pre-process the data by normalizing pixel values (between 0 and 1), reshaping images, and one-hot encoding labels.

3. 

* Design the CNN architecture with convolutional layers, pooling layers, and fully connected layers.
* Use activation functions like ReLU to introduce non-linearity.
* The final layer has 10 units with softmax activation for multiclass classification.

4. 

- Specify the optimizer (e.g., 'adam'), loss function (e.g., 'categorical\_crossentropy' for multiclass classification), and evaluation metric (e.g., 'accuracy').

5. 

* Train the CNN using the training dataset.
* Specify the number of epochs (passes through the entire dataset) and batch size.

6. 

* Evaluate the trained CNN on the test dataset to assess its performance.
* Measure metrics such as accuracy to understand how well the model generalizes to unseen data.

These steps provide a comprehensive overview of the process involved in building and training a CNN for image classification using the MNIST dataset. Adjustments to these steps can be made based on specific model requirements and task objectives.

Recognizes handwritten digits (0-9) with applications in automated systems.

•

•

•



Processes checks by recognizing handwritten amounts and account numbers.

Recognizes postal codes on envelopes for automated mail sorting

* Classifies documents based on handwritten patterns or characters.

Analyzes medical images for detecting anomalies or identifying patterns.

•

•

•

transportation.



Recognizes handwritten characters in forms for efficient data entry

Recognizes hand gestures for sign language translation or device control

Reads and interprets product labels for inventory or quality control.

Identifies handwritten or printed characters on traffic signs for intelligent

* Analyzes historical handwritten documents for digitization and preservation.

CNN excels in diverse applications: finance (check processing), healthcare (image analysis), retail (label recognition), and transportation (sign recognition). It aids digitization efforts and ensures cultural heritage preservation. Overall, CNN offers valuable automation and efficiency across industries.



https://[www.geeksforgeeks.org/applying-convolutional-neural-network-on-mnist-dataset/](http://www.geeksforgeeks.org/applying-convolutional-neural-network-on-mnist-dataset/) https://[www.youtube.com/watch?v=9cPMFTwBdM4](http://www.youtube.com/watch?v=9cPMFTwBdM4)

**Conclusion:** Thus I Build a Multiclass classifier using the CNN model using MNIST or any other suitable dataset by Performing Data Pre-processing, Defining Model and perform training and Evaluating Results using confusion matrix.

**Name: Ashwin Kalekar**

**Roll No: BEAD21143**

**Subject: Computer Laboratory-IV**

**Class : BE**

**Branch: AI & DS**

**ASSIGNMENT No: 03**

**Title:-**

Design RNN or its variant including LSTM or GRU

a) Select a suitable time series dataset. E.g - Predict sentiments based on product reviews. b) Apply for prediction



Implement a Recurrent Neural Network (RNN) or its variant (LSTM or GRU) on a selected time series dataset, such as predicting sentiments based on product reviews, to develop a predictive model for sentiment analysis.

Solve the language translation problem by Recurrent neural network(RNN)



* Python (3.x recommended)
* Jupyter Notebook or any Python IDE or Google Colab



A machine with sufficient RAM and processing power for model training (8GB RAM recommended)



* Basic understanding of Python programming
* Familiarity with the concepts of Neural Networks



Inbuilt tensorflow-keras-imdb dataset



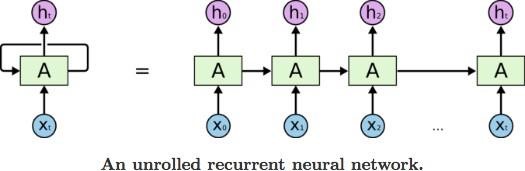
* Keras
* Tensorflow



Recurrent Neural Network (RNN)

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.

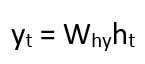


First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step. So, the h(0) and X(1) is the input for the next step. Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training The formula for the current state is

. 

Applying Activation Function:

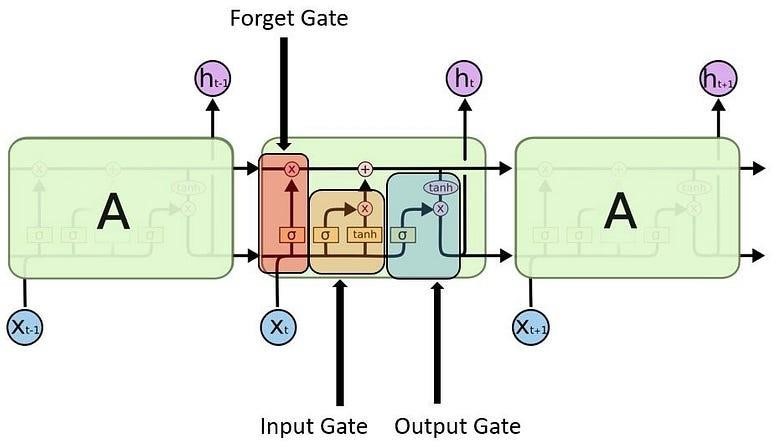


W is weight, h is the single hidden vector, Whh is the weight at previous hidden state, Whx is the weight at current input state, tanh is the Activation funtion, that implements a Non-linearity that squashes the activations to the range[-1.1]

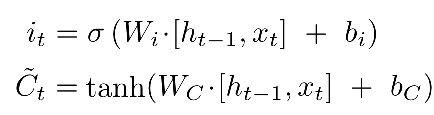
Yt is the output state. Why is the weight at the output state.

Long Short Term Memory (LSTM)

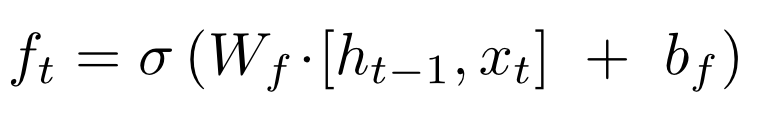
Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present:

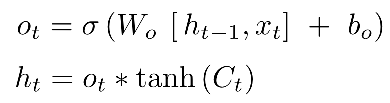


1. Input gate - discover which value from input should be used to modify the memory. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1.

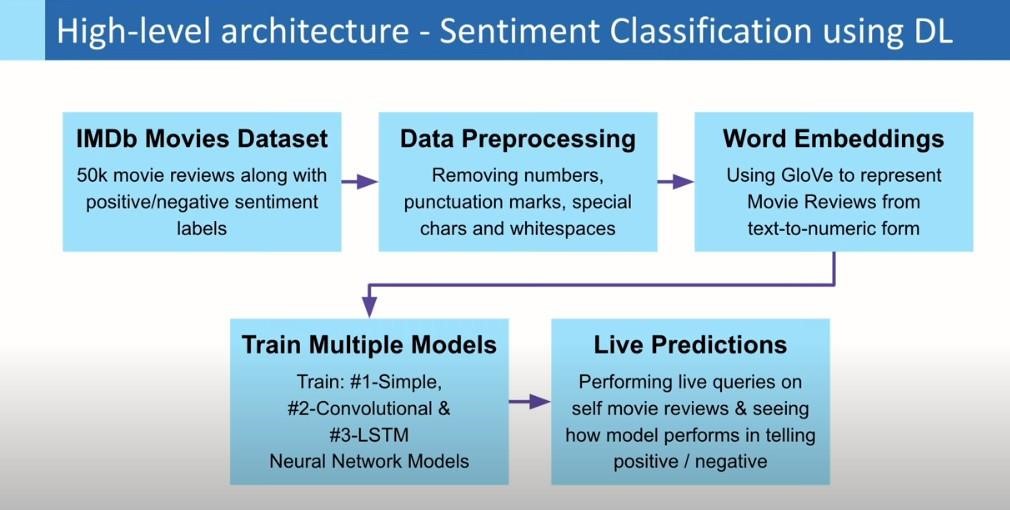


1. Forget gate - discover what details to be discarded from the block. It is decided by the sigmoid function. it looks at the previous state(ht-1) and the content input(Xt) and outputs a number between 0(omit this)and 1(keep this)for each number in the cell state Ct−1.

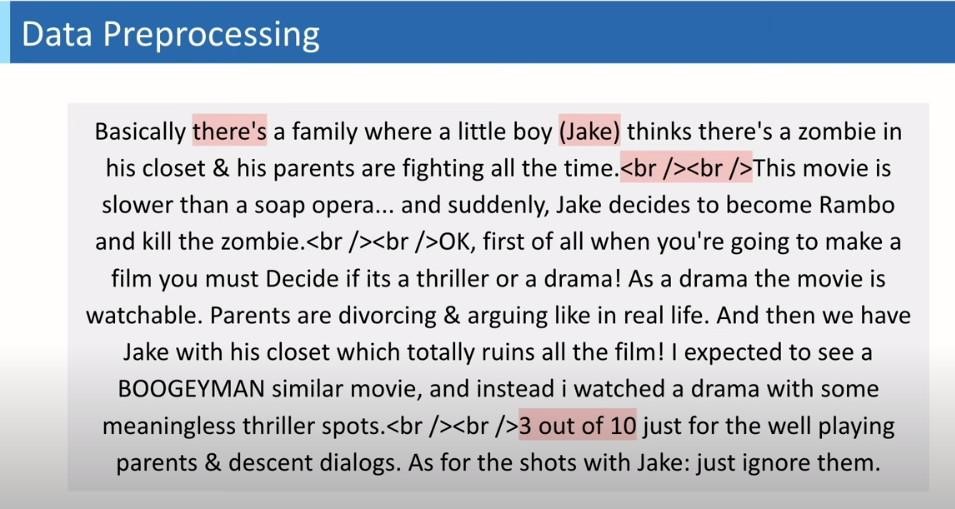


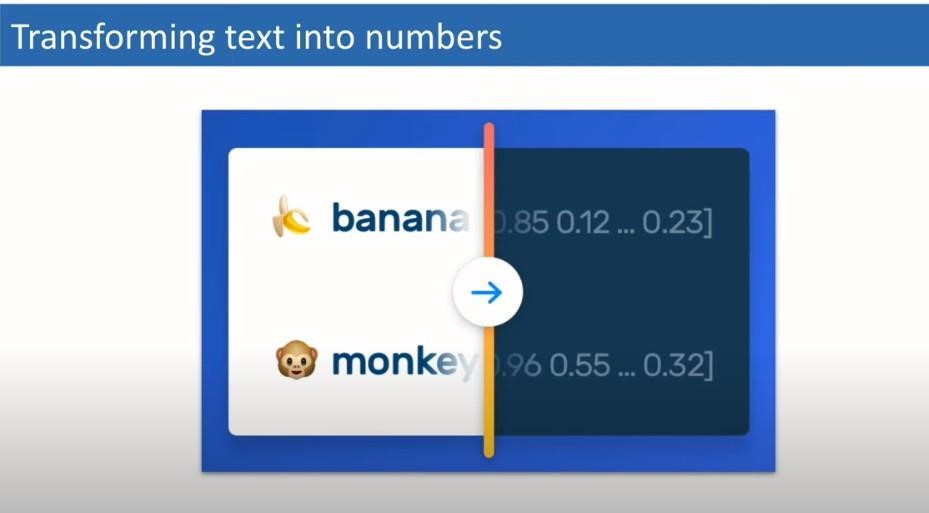
1. Output gate — the input and the memory of the block is used to decide the output. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of Sigmoid.

1. Load IMDb Movie Reviews dataset (50,000 reviews)

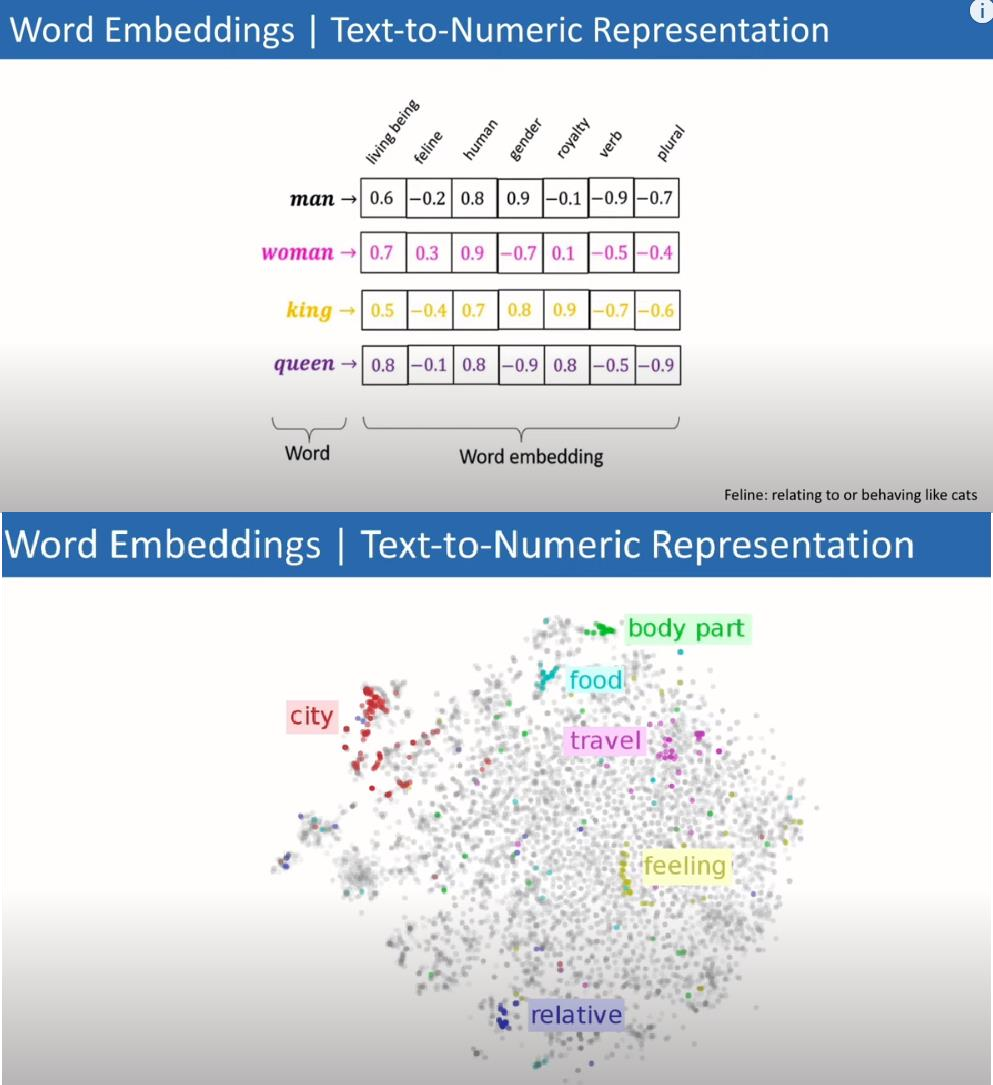


1. Pre-process dataset by removing special characters, numbers, etc. from user reviews + convert sentiment labels positive & negative to numbers 1 & 0, respectively





1. Import GloVe Word Embedding to build Embedding Dictionary + Use this to build Embedding Matrix for our Corpus



1. Model Training using Deep Learning in Keras for separate: Simple Neural Net, CNN and LSTM Models



1) Product Review Sentiment Analysis:

Predict sentiment (positive, negative, neutral) from user reviews for product improvement insights. 2) Customer Feedback Analysis:

Analyze sentiments in customer feedback to understand overall satisfaction and identify areas for improvement. 3) Brand Monitoring:

Monitor social media for product mentions and analyze sentiments to assess brand perception. 4) Market Research:

Analyze sentiments in market surveys to gauge consumer opinions about specific products or features. 5) Quality Assurance in E-commerce:

Automatically categorize and flag reviews with negative sentiments to improve product quality.

The process involves data preparation, embedding, model design, training, and evaluation.

The use of different architectures such as Simple Neural Net, CNN, and LSTM allows for comparison and analysis of their performance on sentiment prediction for IMDb movie reviews. The GloVe Word Embedding enhances the models' understanding of the textual data.

Finally, predictions are made on real IMDb movie reviews to assess the models' applicability and accuracy.



https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e https://colah.github.io/posts/2015-08-Understanding-LSTMs/ https://youtu.be/oWo9SNcyxlI?si=0OzO6SUYZ\_FxbTgY

**Conclusion:** Thus Designed RNN or its variant including LSTM or GRU

**Name: Ashwin Kalekar**

**Roll No: BEAD21143**

**Subject: Computer Laboratory-IV**

**Class : BE**

**Branch: AI & DS**

# ASSIGNMENT No: 07

**Title:** Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.

**Problem Statement:** Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.

**Prerequisite:**

Basics of Python

**Software Requirements:** Power BI Tool

**Hardware Requirements:** PIV, 2GB RAM, 500 GB HDD

**Learning Objectives:**

Learn to import data from different sources such as(Excel, Sql Server, Oracle etc.) and load in targeted system.

**Outcomes:**

After completion of this assignment students are able to understand how to import data from different sources such as(Excel, Sql Server, Oracle etc.) and load in targeted system.

**Theory:**

**What is Legacy Data?**

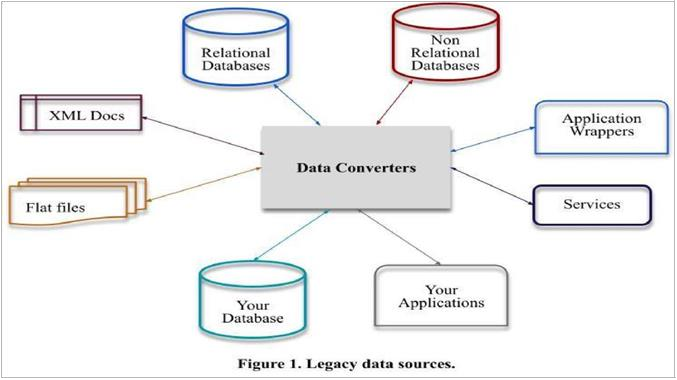
Legacy data, according to Business Dictionary, is "information maintained in an old or outof-date format or computer system that is consequently challenging to access or handle."

**Sources of Legacy Data**

Where does legacy data come from? Virtually everywhere. Figure 1 indicates that there are many sources from which you may obtain legacy data. This includes existing databases, often relational, although non-RDBs such as hierarchical, network, object, XML, object/relational databases, and NoSQL databases. Files, such as XML documents or "flat files"ù such as configuration files and comma-delimited text files, are also common sources of legacy data. Software, including legacy applications that have been wrapped ( perhaps via CORBA) and

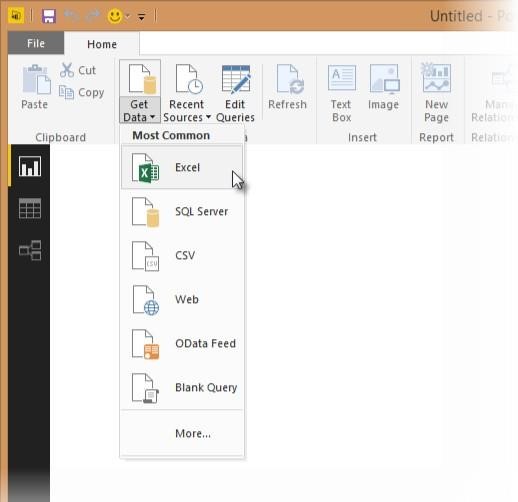
access to existing information. The point to be made is that there is often far more to gaining access to legacy data than simply writing an SQL query against an existing relational database.

legacy services such as web services or CICS transactions, can also provide access to existing information. The point to be made is that there is often far more to gaining



***Importing Excel Data***

1. Launch Power BI Desktop.
2. From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select it directly from the Get Data menu



1. If you select the Get Data button directly, you can also select FIle > Excel and select Connect.

1. In the Open File dialog box, select the Products.xlsx file.

1. In the Navigator pane, select the Products table and then select Edit.

***Importing Data from OData Feed***

In this task, you'll bring in order data. This step represents connecting to a sales system. You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below: <http://services.odata.org/V3/Northwind/Northwind.svc/>Connect to an OData feed:

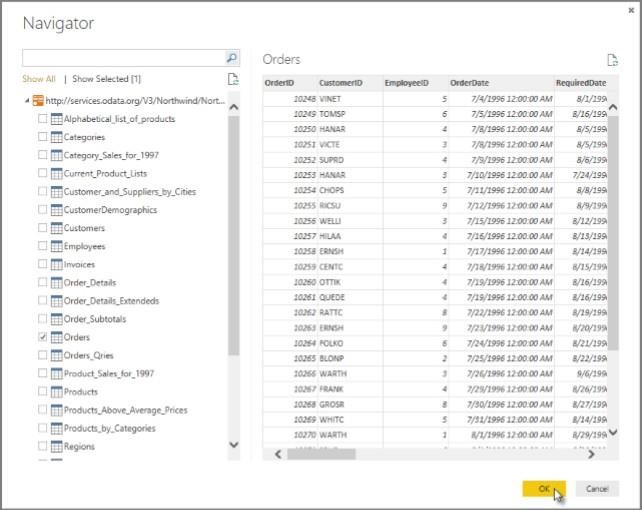
1. rom the Home ribbon tab in Query Editor, select Get Data.

1. Browse to the OData Feed data source.

1. In the OData Feed dialog box, paste the URL for the Northwind OData feed.

1. Select OK.

1. In the Navigator pane, select the Orders table, and then select Edit.



**Conclusion: -** This way, Implemented a program for inverted files.

**Name: Ashwin Kalekar**

**Roll No: BEAD21143**

**Subject: Computer Laboratory-IV**

**Class : BE**

**Branch: AI & DS**

# ASSIGNMENT No: 08

**Title:** Data Visualization from Extraction Transformation and Loading (ETL) Process.

**Problem Statement:** Data Visualization from Extraction Transformation and Loading (ETL) Process.

**Prerequisite:**

Basics of Python

**Software Requirements:**[Jupyter](https://cocalc.com/features/jupyter-notebook)**Hardware Requirements:**

PIV, 2GB RAM, 500 GB HDD

**Outcomes:**

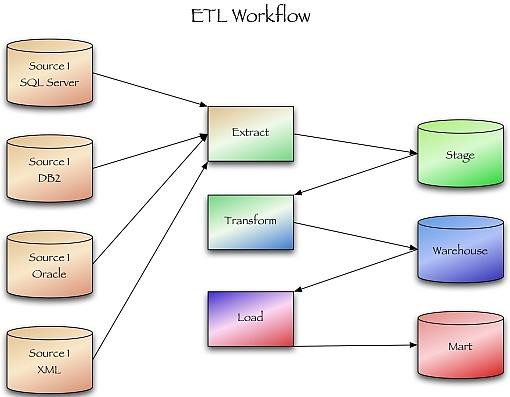
After completion of this assignment students are able to understand how Data Visualization is done through Extraction Transformation and Loading (ETL) Process **Theory:**

Extract, transform, and load (ETL) are 3 data processes, followed after [data collection.](https://apandre.wordpress.com/data/collector/) Extraction takes data, collected in data sources like flat files, databases (relational, hierarchical etc.), transactional datastores, semi-structured repositories (e.g. email systems or document libraries) with different structure and format, pre-validating extracted data and parsing valid data to destination (e.g. staging database)

Transformation takes extracted data and applies predefined rules and functions to it, including selection (e.g.

ignore or remove NULLs), data cleansing, encoding (e.g. mapping “Male” to “M”), deriving (e.g. calculating designated value as a product of extracted value and predefined constant) , sorting, joining data from multiple sources (e.g. lookup or merge), aggregation (e.g. summary for each month), transposing (columns to rows or vice versa), splitting, disaggregation, lookups (e.g. validation through dictionaries), predefined validation etc. which may lead to rejection of some data. Transformed data can be stored into Data Warehouse (DW).

Load takes transformed data and places it into end target, in most cases called Data Mart ( sometimes they called Data Warehouse too). Load can append, refresh or/and overwrite preexisting data, apply constraints and execute appropriate triggers (to enforce data integrity, uniqueness, mandatory fields, provide log etc.) and may start additional processes, like data backup or replication.





**Conclusion:**-

This way Data Visualization from Extraction Transformation and Loading (ETL) Process is done.

**Name: Ashwin Kalekar**

**Roll No: BEAD21143**

**Subject: Computer Laboratory-IV**

**Class : BE**

**Branch: AI & DS**

# ASSIGNMENT No: 09

**Title:** Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

**Problem Statement:** Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

**Prerequisite:**

Basics of Python

**Software Requirements:**[Jupyter](https://cocalc.com/features/jupyter-notebook)**Hardware Requirements:**

PIV, 2GB RAM, 500 GB HDD

**Outcomes:**

After completion of this assignment students are able to understand how to Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

**Theory:**

**Step 1 : Data Extraction :**

The data extraction is first step of ETL. There are 2 Types of Data Extraction

Full Extraction : All the data from source systems or operational systems gets extracted to staging area. (Initial Load)

Partial Extraction : Sometimes we get notification from the source system to update specific date. It is called as Delta load.

Source System Performance: The Extraction strategies should not affect source system performance.

**Step 2 : Data Transformation :**

The data transformation is second step.After extracting the data there is big need to do the transformation as per the target system.I would like to give you some bullet points of Data Transformation.

* + Data Extracted from source system is in to Raw format. We need to transform it before loading in to target server.
  + Data has to be cleaned, mapped and transformed.
  + There are following important steps of Data Transformation :

1. **Selection :** Select data to load in target

1. **Matching :** Match the data with target system

1. **Data Transforming :** We need to change data as per target table structures
   * Character set conversion : Need to transform the character sets as per the target systems. (Firstname and last name example)
   * Calculated and derived values: In source system there is first val and second val and in target we need the calculation of first val and second val.
   * Data Conversion in different formats : If in source system date in in DDMMYY format and in target the date is in DDMONYYYY format then this transformation needs to be done at transformation phase.

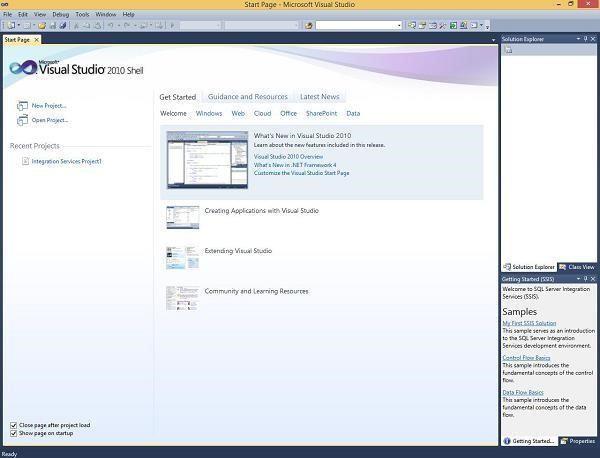
## Step 3 : Data Loading

• [Data loading p](https://www.vskills.in/certification/tutorial/data-mining-and-warehousing/data-loading-types-and-modes/)hase loads the prepared data from staging tables to main tables.

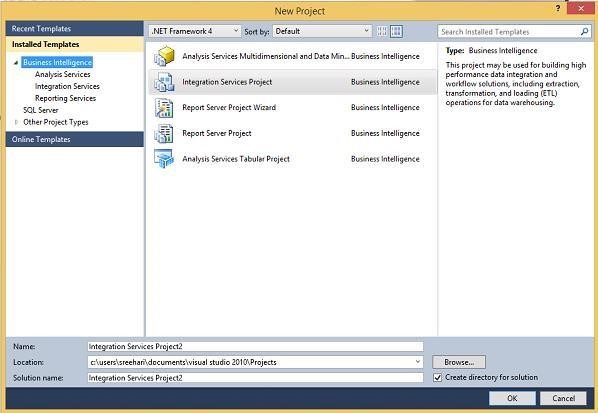
ETL process in SQL Server:

Following are the steps to open BIDS\SSDT.

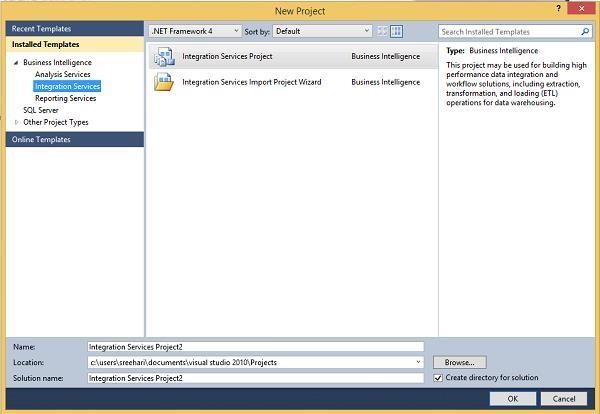
**Step 1** − Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group. The following screen appears



**Step 2** − The above screen shows SSDT has opened. Go to file at the top left corner in the above image and click New. Select project and the following screen opens.



**Step 3** − Select Integration Services under Business Intelligence on the top left corner in the above screen to get the following screen.



**Step 4** − In the above screen, select either Integration Services Project or Integration Services Import Project Wizard based on your requirement to develop\create the package.

There are two modes − Native Mode (SQL Server Mode) and Share Point Mode.

There are two models − Tabular Model (For Team and Personal Analysis) and Multi Dimensions Model (For Corporate Analysis).

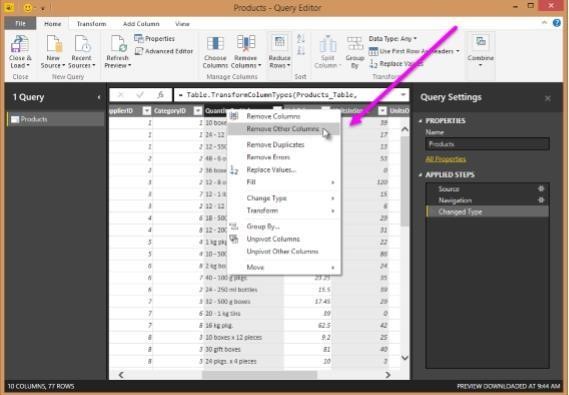
The BIDS (Business Intelligence Studio till 2008 R2) and SSDT (SQL Server Data Tools from 2012) are environments to work with SSAS.

## ETL Process in Power BI

**1) Remove other columns to only display columns of interest**

Power BI Desktop includes Query Editor, which is where you shape and transform your data connections. Query Editor opens automatically when you select **Edit** from Navigator. You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

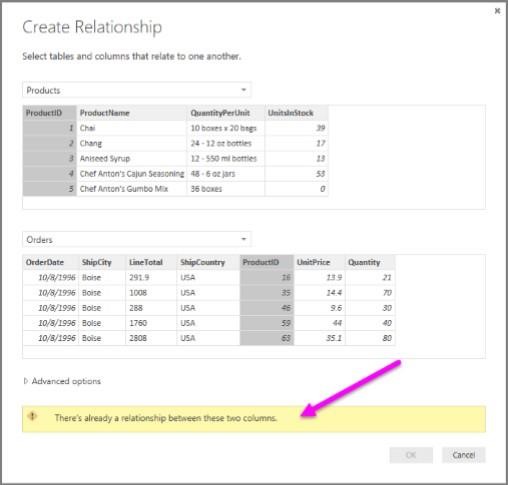
1. In **Query Editor**, select the **ProductID, ProductName**, **QuantityPerUnit,** and **UnitsInStock** columns (use **Ctrl+Click** to select more than one column, or **Shift+Click** to select columns that are beside each other).
2. Select **Remove Columns > Remove** Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.



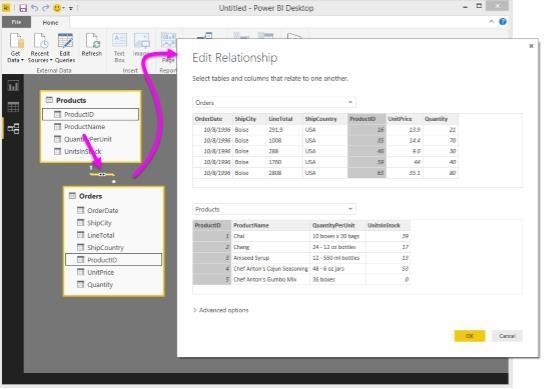
1. **Change the data type of the UnitsInStock column**

When Query Editor connects to data, it reviews each field and to determine the best data type. For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the **UnitsInStock** column’s datatype is Whole Number.

1. Select the **UnitsInStock** column.
2. Select the **Data Type drop-down button** in the **Home ribbon**.
3. If not already a Whole Number, select **Whole Number** for data type from the drop down (the Data Type: button also displays the data type for the current selection).
4. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.



1. Select Cancel, and then select Relationship view in Power BI Desktop.
2. When you double-click the arrow on the line that connects the to queries, an Edit Relationship dialog appears.



**Conclusion:** Thus Performed Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.