Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Program in
Machine Learning and Artificial Intelligence

Building A Conversational Chatbot

Capstone Project - I: AI-ML PG Program by NITW E&ICT Academy

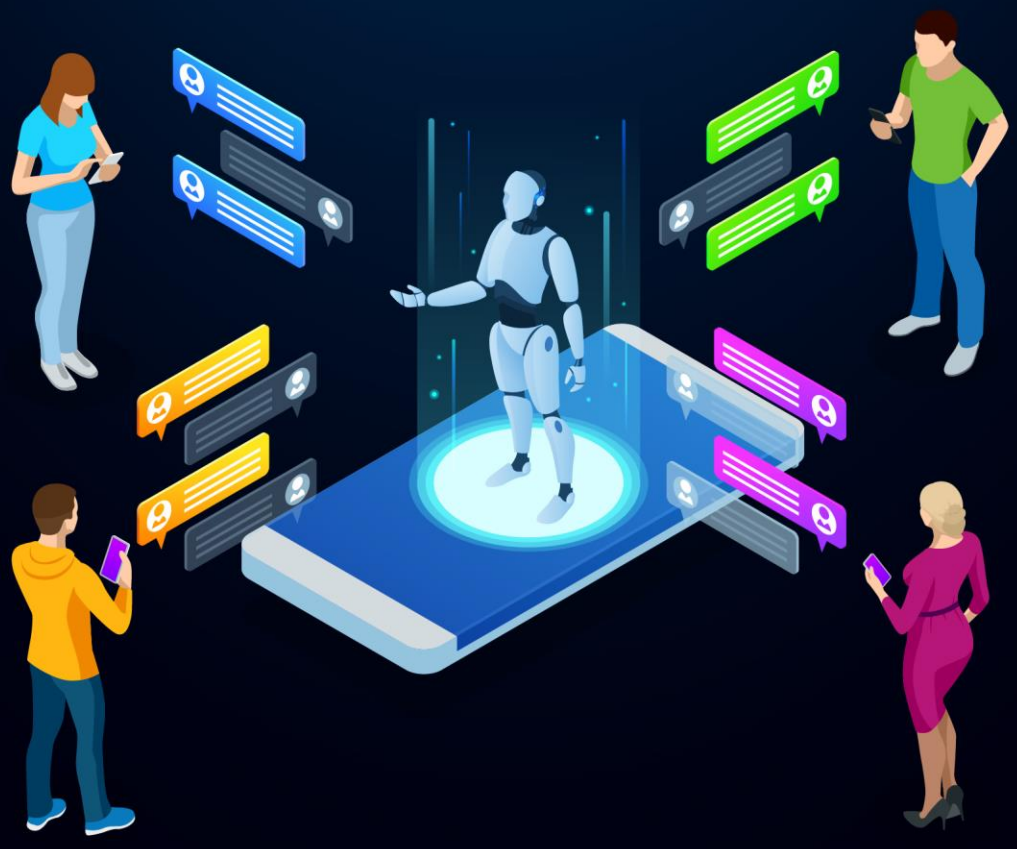


Table of Contents

Aim of the Project

Aim of the project is to build an intelligent conversational chatbot, **Riki**, that can understand complex queries from the user and intelligently respond.

Background

R-Intelligence Inc., an AI startup, has partnered with an online chat and discussion website bluedit.io. They have an average of over 5 million active customers across the globe and more than 100,000 active chat rooms. Due to the increased traffic, they are looking at improving their user experience with a chatbot moderator, which helps them engage in a meaningful conversation and keeps them updated on trending topics, while merely chatting with Riki, a chatbot. The Artificial Intelligence-powered chat experience provides easy access to information and a host of options to the customers.

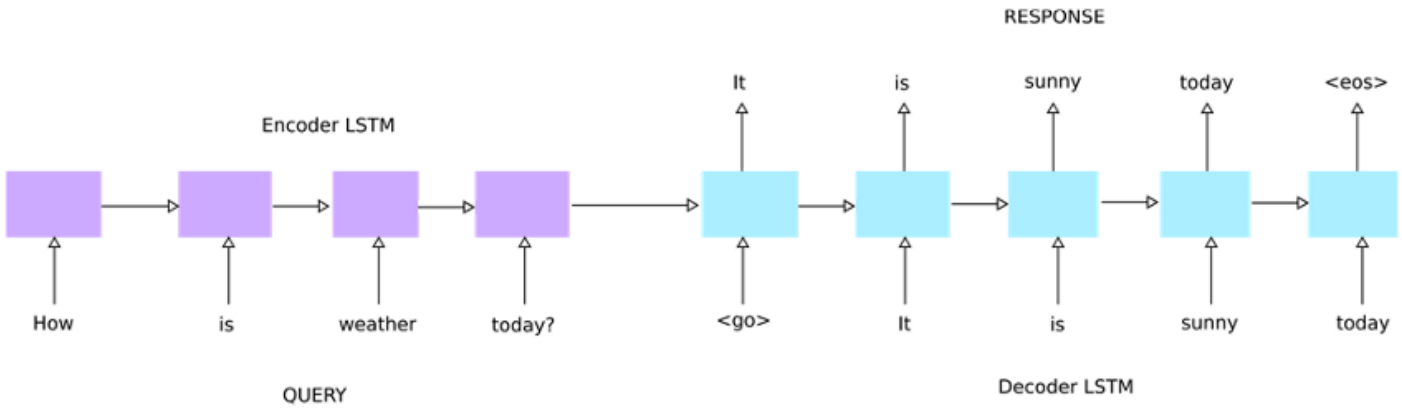
Business Requirement

R-Intelligence Inc. has invested in Python, PySpark, and Tensorflow. Using emerging technologies of Artificial Intelligence, Machine Learning, and Natural Language Processing, **Riki – the chatbot** should make the whole conversation as realistic as talking to an actual human.

The chatbot should understand that users have different intents and make it extremely simple to work around these by presenting the users with options and recommendations that best suit their needs.

Suggested Approach

R-Intelligence Inc. used an approach using only Natural Language Processing, in which Seq2seq models (encoder and Decoder) are used as the state-of-the-art approach to implement end to end text generation for a conversational bot.



Tasks to be performed

- Download the glove model available at https://nlp.stanford.edu/projects/glove/

  Specification: Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Load the glove word embedding into a dictionary where the **key** is a unique **word token** and the value is a **d** dimension vector
- **Data Preparation** - Filter the conversations till max word length and convert the dialogues pairs into input text and target texts. Put **start** and **end** token to recognize the beginning and end of the sentence token
- Create two dictionaries:
  - target_word2id
  - target_id2word

  and save it as NumPy file format in the disk.
- Prepare the input data with embedding. The input data is a list of lists:
  - First list is a list of sentences
  - Each sentence is a list of words
- Generate training data per batch
- Define the model architecture and perform the following steps:
  - **Step 1:** Use a LSTM encoder to get input words encoded in the form of (encoder outputs, encoder hidden state, encoder context) from input words
  - **Step 2:** Use a LSTM decoder to get target words encoded in the form of (decoder outputs, decoder hidden state, decoder context) from target words. Use encoder hidden states and encoder context (represents input memory) as initial state.

- **Step 3:** Use a dense layer to predict the next token out of the vocabulary given decoder output generated by Step 2.
- **Step 4:** Use loss ='categorical_crossentropy' and optimizer='rmsprop'

- Generate the model summary
- Finally generate the prediction

Dataset Description

Dataset: Cornell Movie Dialogue corpus

Brief Description

This corpus contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts:

- 220,579 conversational exchanges between 10,292 pairs of movie characters
- involves 9,035 characters from 617 movies
- in total 304,713 utterances
- movie metadata included:
  - genres
  - release year
  - IMDB rating
  - number of IMDB votes
  - IMDB rating
- character metadata included:
  - - gender (for 3,774 characters)
  - - position on movie credits (3,321 characters)

File Description

In all files the field separator is " +++$+++ "

- movie_titles_metadata.txt

  Contains information about each movie title

  - fields:
    - movieID,
    - movie title,
    - movie year,
    - IMDB rating,
    - no. IMDB votes,
    - genres in the format ['genre1','genre2',É,'genreN']

- movie_characters_metadata.txt

  Contains information about each movie character

  - fields:
    - characterID
    - character name
    - movieID
    - movie title
    - gender ("?" for unlabeled cases)
    - position in credits ("?" for unlabeled cases)

- movie_lines.txt

  Contains the actual text of each utterance

  - fields:
    - lineID
    - characterID (who uttered this phrase)
    - movieID
    - character name
    - text of the utterance

- movie_conversations.txt

Contains the structure of the conversations

- fields
  - *characterID* of the first character involved in the conversation
  - *characterID* of the second character involved in the conversation
  - *movieID* of the movie in which the conversation occurred
  - List of the utterances that make the conversation, in chronological order: ['lineID1','lineID2',É,'lineIDN'] has to be matched with movie_lines.txt to reconstruct the actual content

- raw_script_urls.txt

  Contains the urls from which the raw sources were retrieved

How to Start with the Project?

1. Login to the Google Co-lab, load the notebook to the environment. Go to Runtime to choose the “Change runtime type”. For faster training, choose GPU as the hardware accelerator and SAVE it.

2. Open the ‘Chatbot.ipynb’ notebook and start filling the code.

3. Import all the necessary Python packages. Numpy and Pandas for numerical processing, data importing, preprocessing etc. Sklearn for splitting datasets, keras/tensorflow for deep learning model creation, training, testing, inference etc.

4. From here you can take over to the project and start building the conversational chatbot.

How to submit your project?

Following are the tasks, which need to be developed while executing the project:

- Share your project via google colab to support@edureka.co
- The .ipynb file with details of each step in the markdown
- You can even upload your code into your github repository and share your repository with us

Marks Allocation

1. Load the glove word embedding into a dictionary. [10 Marks]
2. Create a target word to id dictionary called target_word2idx. [10 Marks]
3. Create a target to id dictionary called target_idx2word. [10 Marks]
4. Prepare the input data with embedding. [15 Marks]
5. Generate training data per batch. [15 Marks]
6. Defining correct model architecture. [10 Marks]
7. Final prediction. [20 Marks]
8. Project report/synopsis with detailed .ipynb (With best Markdown explanation) . [10 Marks]

Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Program in

Machine Learning and Artificial Intelligence

Action recognition using pose estimation

Capstone Project II AI ML PG Program by NITW E&ICT Academy

Table of Contents

Aim of the Project

Aim of this project is to automatically recognize human actions based on analysis of the body landmarks using pose estimation.

Learning Outcome

1. Implementation of Convolutional Neural Network based pose estimation for body landmark detection
2. Implementation of pose features-based action recognition and its improvement using graphical feature representation and data augmentation of body landmarks
3. Preparation and preprocessing of image datasets
4. Fine tuning and improvement of the action recognition model with better feature representation and data augmentation
5. Development, error analysis and deep learning model improvement

Problem Statement

Analysis of people's actions and activities in public and private environments are highly necessary for security. This cannot be done manually as the number of cameras for surveillance produce lengthy hours of video feed every day. Real-time detection and alerting of suspicious activities or actions are also challenging in these scenarios. This issue can be solved by applying deep learning-based algorithms for action recognition.

Project Description

The project has 3 major components:

1. Implementation of CNN based Pose Estimation model
2. Implementation of NN based Action Recognition model
3. Implementation of Action Recognition in videos using pose joints estimated by the CNN model

1. CNN based Pose Estimation model

Unpack and load the pre-processed the dataset with images and joint coordinates. Each pose image contains 7 body joints represented using 14 floating point numbers. Build the CNN model for the dataset and train for few epochs. Test the model with testing set and do inference with single data samples. Save the model. Train a transfer learning model with already trained CNN base model.

2. NN based Action Recognition model

Load the dataset with pose joints as features and actions labels ('Namaste' vs 'Hello'). Split the dataset into training and testing. Train the model for few epochs, test the model, save the model. Analyze the model with new samples of data. Based on the results, flip invariant data augmentation is performed to increase the accuracy where further graph-like features are

extracted from the dataset. Training, testing and inferencing is accomplished with the augmented graph features of the dataset.

3. Action Recognition in videos

CNN based pose estimation model and neural networks-based action recognition model is combined to work on images and videos. Action recognition in videos is achieved by processing the frames one by one using the model.

Methodology

Pose Estimation

Input to the human pose estimation model is closely cropped pedestrian or human image. Each training set image has one human inside where pixels are considered as features and target as pair of body joint coordinates. Pre-processed Pose Estimation FLIC dataset (http://bensapp.github.io/flic-dataset.html) is be used for this modelling. Set of landmarks from human image can be detected by training convolutional neural networks model with convolution, pooling and fully connected layers with finally landmark point regression as output. Model can be trained based on two strategies:

Strategy 1:

1. Training a model from scratch where model layers can be designed manually, weights of the model will be trained
2. Loading a pretrained base CNN architecture such as VGG16, MobileNet, removing the final softmax layer adding custom layers and training the newly added layers.

Strategy 2:

This strategy makes use of “transfer learning” where we can choose whether to retrain the whole network or train only newly added layers. Model can be inferred using a single test sample which is not part of the training/test set, detected pose points can be further plotted.

Action Recognition

Set of human actions can be further recognized using analysis of pose landmarks as features and action label as target. Deep neural networks can be directly trained using dense layers by considering pose point x and y coordinates. A custom dataset with two actions - Namaste, Hello will be used for this task.

Action recognition neural network can be built by directly considering x and y coordinates as feature and action label as target. Since human can appear in any scale in a real scenario, considering raw coordinates as features is not a good idea. This can be solved by considering the skeleton of human pose points as graph, extracting the distance between every joint and further normalizing the distance features. These distance features can be considered for training the action recognition model.

Dataset contains Hello actions performed using right-hand only. Hence, the model cannot recognize a Hello human action performed with left-hand. This issue can be solved by augmenting the action dataset by duplicating the existing actions further flipping the coordinates horizontally. Now with the augmented action dataset, the trained model can detect Hello action performed in both right as well as left-hand. Finally, the pose estimation and action recognition models can be combined by running sequentially. The model can be tested with offline videos and action recognition results will be displayed for each video frame.

Expected Challenges

The project will aim at tackling the following technical challenges:

1. Body landmarks detection of humans from different viewpoints.
2. Detection of human actions which involves pose detection of humans appearing at any scale in the video frame with action performed using either left or right or both body parts.
3. System must meet the real-time processing requirements where the deep learning model must detect pose and actions from video (with 30 fps) at the rate of 33 ms per frame.

Dataset Description

Dataset: Frames Labeled in Cinema (FLIC)

The dataset is a collection of 5003 image from popular Hollywood movies. The images were obtained by running a state-of-the-art person detector on every tenth frame of 30 movies. People detected with high confidence (roughly 20K candidates) were then sent to the crowdsourcing marketplace Amazon Mechanical Turk to obtain ground-truth-labeling. Each image was annotated by five Turkers for $0.01 each to label 10 upper body joints. The median-of-five labeling was taken in each image to be robust to outlier annotation. Finally, images were rejected manually by us if the person was occluded or severely non-frontal. We set aside 20% (1016 images) of the data for testing.

How to Start with the Project?

1. Login to the Google Co-lab, load the notebook to the environment. Go to Runtime to choose the "Change runtime type". For faster training, choose GPU as the hardware accelerator and SAVE it.

2. Import all the necessary Python packages. Numpy and Pandas for numerical processing, data importing, preprocessing etc. Matplotlib for plotting pose joints and showing images, cv2 package for image processing functions, sklearn for splitting datasets, keras for deep learning model creation, training, testing, inference etc.

3. Dataset for Pose Estimation and Action Recognition can be kept inside Google Drive and can be loaded to the Colab. Dataset in the google drive can be accessed using the path "gdrive/My Drive/Dataset_Folder_Name/Dataset_Name.zip". The pose dataset used in this project is pre-processed FLIC (http://bensapp.github.io/flic-dataset.html) where the Action Recognition dataset is custom made. These datasets can be downloaded and uploaded to the google colab current working directory or it can be kept in google drive which can be mounted to the Colab working directory.

4. Training images and their annotations in action_joints.csv consists of 7 joints - 'left shoulder', 'left elbow', 'left wrist', 'right shoulder', 'right elbow', 'right wrist', 'left eye', 'right eye', 'nose'. The files are read and converted into numpy arrays so that the numpy arrays can be used for training the model. Since the dataset does not provide the validation set (Validation set helps us to monitor the training after each epoch) part of training set can be considered as validation set using function train_test_split. The parameter test_size is the ration between number of training and validation set samples, can be set based on the how bigger we want the validation set to the compared to the training set.
5. From here you can take over to the project and start estimating the human pose.

How to submit your project?

Following are the tasks, which need to be developed while executing the project:

- Share your project via google colab to support@edureka.co
- The .ipynb file with details of each step in the markdown
- Save the model and mail the .h5 created and the downloaded .ipynb to support@edureka.co
- You can even upload your code into your github repository and share your repository with us

Marks Allocation

1. Implementation of CNN based Pose Estimation model [20 Marks]
2. Implementation of NN based Action Recognition model [20 Marks]
3. Implementation of Action Recognition in videos using pose joints estimated by the CNN [20 Marks]
4. Project report/synopsis with detailed .ipynb [15 Marks]
5. Testing your model on the evaluation dataset [25 Marks]

Note: Evaluation dataset won't be shared with you.

Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Program in
Machine Learning and Artificial Intelligence

ML Model for Auto Insurance Industry

Capstone Project - III: AI-ML PG Program by NITW E&ICT Academy

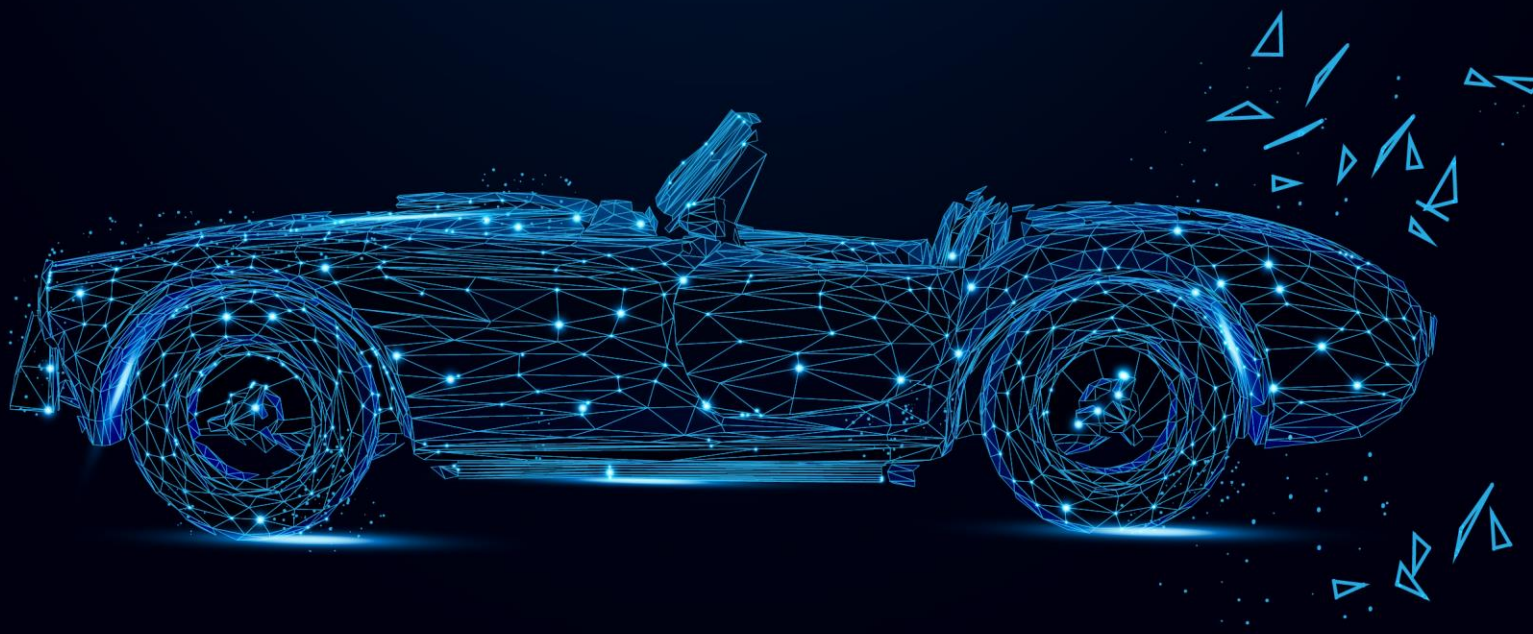


Table of Contents

Aim of the Project

The aim of the project is to build a Machine Learning Model to predict whether an owner will initiate an auto insurance claim in the next year.

Background

The auto insurance industry is witnessing a paradigm shift. Since auto insurance company consists of homogenous good thereby making it difficult to differentiate product A from product B, also companies are fighting a price war (for insurance price). On top of that, the distribution channel is shifting more from traditional insurance brokers to online purchases, which means that the ability for companies to interact through human touchpoints is limited, and customers should be quoted at a reasonable price. A good price quote is one that makes the customer purchase the policy and helps the company to increase the profits.

Also, the insurance premium is calculated based on more than 50+ parameters, which means that traditional business analytics-based algorithms are now limited in their ability to differentiate among customers based on subtle parameters.

Use Cases

The model shall mainly support the following use cases:

1. Conquering Market Share: Capture market share by lowering the prices of the premium for the customers, who are least likely to claim.

2. Risk Management: Charge the right premium from the customer, who is likely to claim insurance in the coming year

3. Smooth Processing: Reduce the complexity of pricing models. Most of the transactions are happening online with larger customer attributes (thanks to the internet and social media). Harness the power of huge data to build complex ML models

4. Increased Profits: As per industry estimate 1% reduction in the claim can boost profit by 10%. So, through the ML model, we can identify and deny the insurance to the driver who will make a claim. Thus, ensuring reduced claim outgo and increased profit.

Part of the model development is to identify and prioritize the above use cases.

Process Flow

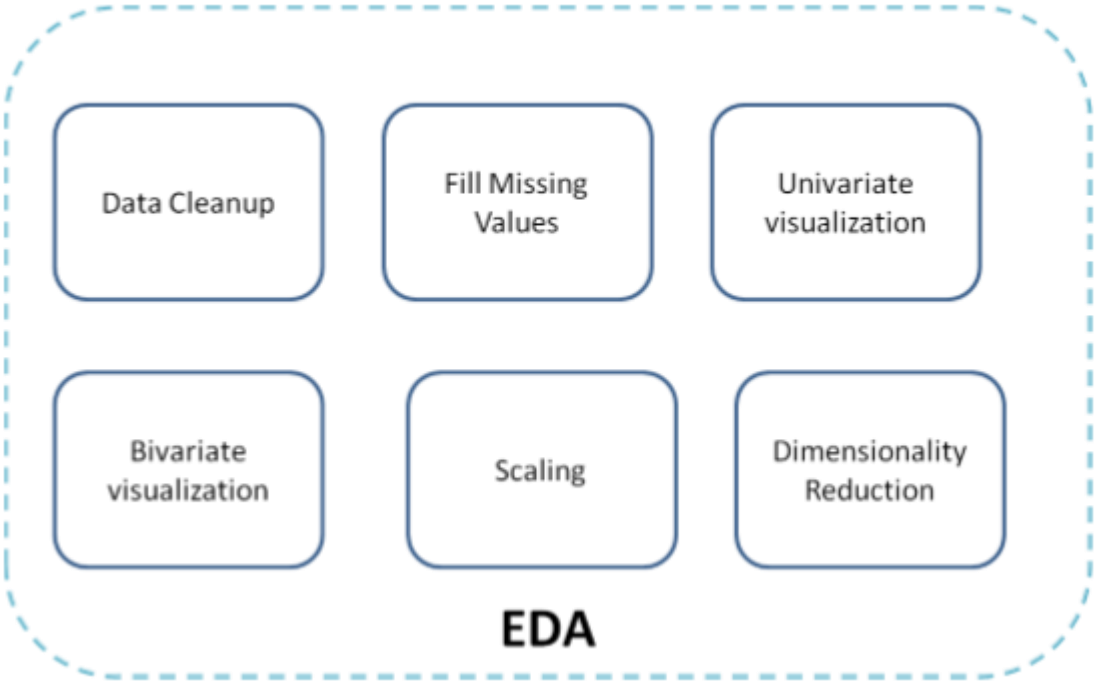
The Machine Learning model mainly consist of two phases:

1. EDA (Exploratory Data Analysis):

Analyze the datasets to summarize their main characteristics (with visual methods). A statistical model can be used, primarily EDA can be used to see what the data can tell us beyond the formal modeling or hypothesis testing task.

Following tasks can be performed as a part of EDA:

- Scaling/Normalization
- Fill the missing values
- Feature selection & engineering

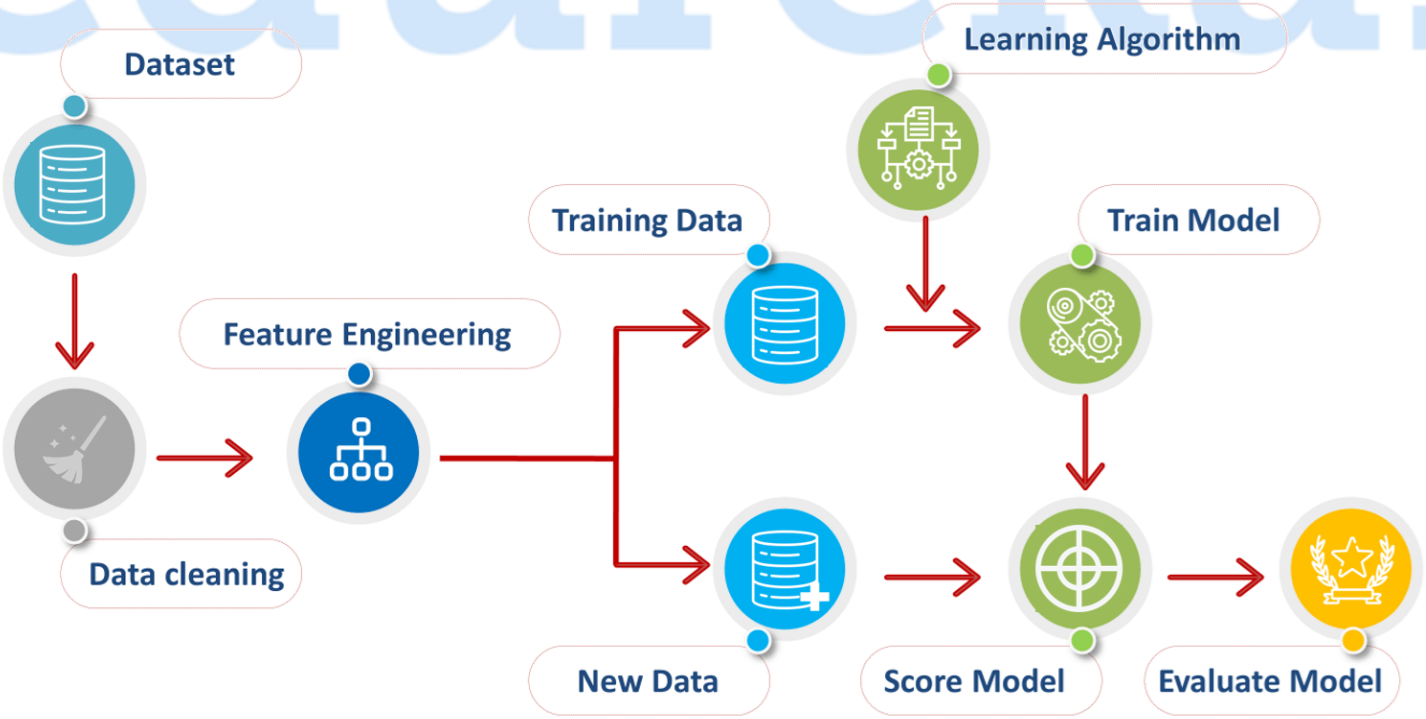


2. Machine Learning Modeling:

After EDA, the modeling comes into the process. The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data. The term "ML model" refers to the model artifact that is created by the training process.

The training data must contain the correct answer (target or target attribute). The learning algorithm finds patterns in the training data that maps the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

You will use the ML model to get predictions on new data for which you will not know the target.



Following tasks can be performed as a part of Modeling:

- Start with the basic model but eventually move towards ensemble
- Use Deep Learning with sklearn MLPClassifier and check if the Neural Network Model is better than traditional models
- Arrival at a model with best f1-score

Dataset Description

The project involves the use of a dataset with 600k training data and 57 features/data. In the train and test data, features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc). In addition, feature names include the postfix bin to indicate binary features and cat to indicate categorical features. Features without these designations are either continuous or ordinal. Values of -1 indicate that the feature was missing from the observation. The target column signifies whether a claim was filed for that policy holder.

Tasks to be performed

Following are the deliverables (.ipynb files), which needed to be developed with respect to Exploratory Data Analysis:

1. Write at least 3 important inferences from the data above
2. Is the data balanced? Meaning are targets 0 and 1 in the right proportion?
3. How many categorical features are there?
4. How many binary features are there?
5. Write inferences from data on interval variables.
6. Write inferences from data on ordinal variables.
7. Write inferences from data on binary variables.

8. Check if the target data is proportionate or not. Hint: Below than 30% for binary data is sign of imbalance
9. What should be the preferred way in this case to balance the data?
10. How many training records are there after achieving a balance of 12%?
11. Which are the top two features in terms of missing values?
12. In total, how many features have missing values?
13. What steps should be taken to handle the missing data?
14. Which interval variables have strong correlation?
15. What's the level of correlation among ordinal features?
16. Implement Hot Encoding for categorical features
17. In nominal and interval features, which features are suitable for StandardScaler?
18. Summarize the learnings of ED

Following are the deliverables (.ipynb files), which needed to be developed with respect to Modeling :

1. The Simple Logistic Regression Model seems to have high accuracy. Is that what we need at all? What is the problem with this model?
2. Why do you think f1-score is 0.0?
3. What is the precision and recall score for the model?
4. What is the most important inference you can draw from the result?
5. What is the accuracy score and f1-score for the improved Logistic Regression model?
6. Why do you think f1-score has improved?

7. For model LinearSVC play with parameters – dual, max_iter and see if there is any improvement

8. SVC with Imbalance Check & Feature Optimization & only 100K Records → is there improvement in scores?

9. XGBoost is one the better classifiers -- but still f1-score is very low. What could be the reason?

10. What is the increase in number of features after one-hot encoding of the data?

11. Is there any improvement in scores after encoding?

12. If not missing a positive sample is the priority which model is best so far?

13. If not marking negative sample as positive is top priority, which model is best so far?

14. Do you think using AdaBoost can give any significant improvement over XGBoost?

15. MLPClassifier is the neural network we are trying. But how to choose the right no. of layers and size?

16. At what layer size we get the best f1-score?

How to Start with the Project?

1. Login to the Google Co-lab, load the notebook to the environment. Go to Runtime to choose the "Change runtime type". For faster training, choose GPU as the hardware accelerator and SAVE it.

2. Import all the necessary Python packages. Numpy and Pandas for numerical processing, data importing, preprocessing etc. Matplotlib for data visualization and performing exploratory data analysis, sklearn for splitting datasets and creating machine learning models, keras/tensorflow for deep learning model creation, training, testing, inference etc.

3. From here you can take over to the project and start building the machine learning model for auto-insurance company

How to submit your project?

Following are the tasks, which need to be developed while executing the project:

- Share your project via google colab to support@edureka.co
- The .ipynb file with details of each step in the markdown
- You can even upload your code into your github repository and share your repository with us

Marks Allocation

1. Exploratory data Analysis [18 x 2 Marks]
2. Modeling [16 x 3 Marks]
3. Project report/synopsis with detailed .ipynb(With best Markdown explanation) [16 Marks]