**TOPIC**

**Telecom Customer Churn Prediction Using Watson Auto AI**

**Name -Muskan Gupta**

**Email- [muskangupta2410@gmail.com](mailto:muskangupta2410@gmail.com)**

**Mobile no.- 9575236666**

**INDEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | |  | **INTRODUCTION** | |  |  |  |
|  | |  | * 1. Overview | |  |  |  |
|  | |  | 1.2 Purpose | |  |  |  |
| **2** | |  | **LITERATURE SURVEY** | |  |  |  |
|  | |  | 2.1 Existing problem | | |  |  |
|  | |  | 2.2 Proposed solution | | |  |  |
| **3** | |  | **THEORITICAL ANALYSIS** | |  |  |  |
|  | |  | 3.1 Block diagram | |  |  |  |
|  | |  | 3.2 Hardware / Software designing | | |  |  |
| **4** | |  | **EXPERIMENTAL INVESTIGATIONS** | | |  |  |
| **5** | |  | **FLOWCHART** | |  |  |  |
| **6** | |  | **RESULT** |  |  |  |  |
| **7** | |  | **ADVANTAGES & DISADVANTAGES** | | |  |  |
| **8** | |  | **APPLICATIONS** | |  |  |  |
| **9** | |  | **CONCLUSION** | |  |  |  |
| **10** | |  | **FUTURE SCOPE** | | **`** |  |  |
| **11** | |  | **BIBILOGRAPHY** | |  |  |  |
|  | |  | **APPENDIX** | |  |  |  |
|  |  | | A. Source code | |  |  |  |
|  |  | |  |  |  |  |  |

**1. INTRODUCTION**

1.1 Overview-

Customer churn is a major problem and one of the most important concerns for large companies.  Telecommunication industry always suffers from very high churn rates when one industry offers a better plan than the previous there is a high possibility of the customer churning from the present due to a better plan in such a scenario it is very difficult to avoid losses but through prediction, we can keep it to a minimal level.

1.2 Purpose-

Due to the direct effect on the revenues of the companies, companies are seeking to develop means to predict potential customers to churn. Therefore, finding factors that increase customer churn is important to take necessary actions to reduce it.

1. **LITERATURE SURVEY**
   1. Existing problem-

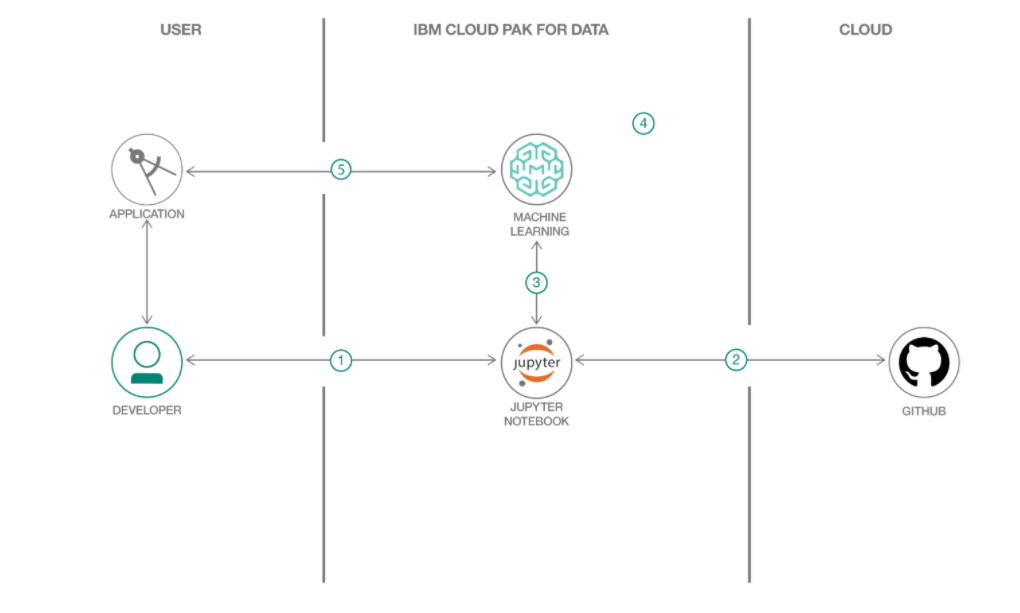
A company experiences an overall of 30% annual churn rate and it is very expensive for the company to recruit new customers. Therefore, the strategy followed by the service providers is to maintain the already existing customers, rather than recruiting new ones. Through this study, if the telecommunications companies know which users are at high risk of churning and when they will churn, the service providers will be able to design customized customer plans and treatment programs for a set of customers in a timely efficient manner. Many aspects may influence a customer to churn. Prepaid customers are not constrained by service contracts and so they churn for this reason. Hence, it is difficult to predict their churn rate. Another factor is customer reliability and sincerity, which can be determined by service and outcome quality given by the telecom industry. Issues like network coverage and reception quality may influence customers to move to the co-rival companies. Other factors that increase probability of customers churning include slow or faulty response to complaints and billing errors. Factors such as encasing prices, inadequate features, and primitive technology may also cause customers to switch to the competitors. Customers often compare their providers with others and churn to whoever they feel provides better overall value for money .

* 1. Proposed solution-

Churn prediction helps in identifying those customers who are likely to leave a company. The main contribution of our work is to develop a churn prediction model which assists telecom operators to predict customers who are most likely subject to churn.

Build & Deploy a Machine Learning model to predict the customer churn using IBM Watson Studio and predictions can be obtained by using its Endpoint. Create a python - flask application that interacts with the model.

1. **THEORITICAL ANALYSIS**
   1. Block diagram-



* 1. Hardware / Software designing-

There was no hardware designing as such. Using IBM Watson Studio I could build the model using Auto AI feature and deploy it there itself. Using VS Code Editor I could build my UI in HTML and connect it to the backend model using Flask.

1. **EXPERIMENTAL INVESTIGATIONS**

While using IBM Watson Studio, I compared the performance of the model using various algorithms and found that Gradient Boosting Classifier algorithm works best. The model uses Hyper Parameter Optimization and Feature Extraction for further enhancement.

1. **FLOWCHART**

Deploy the model

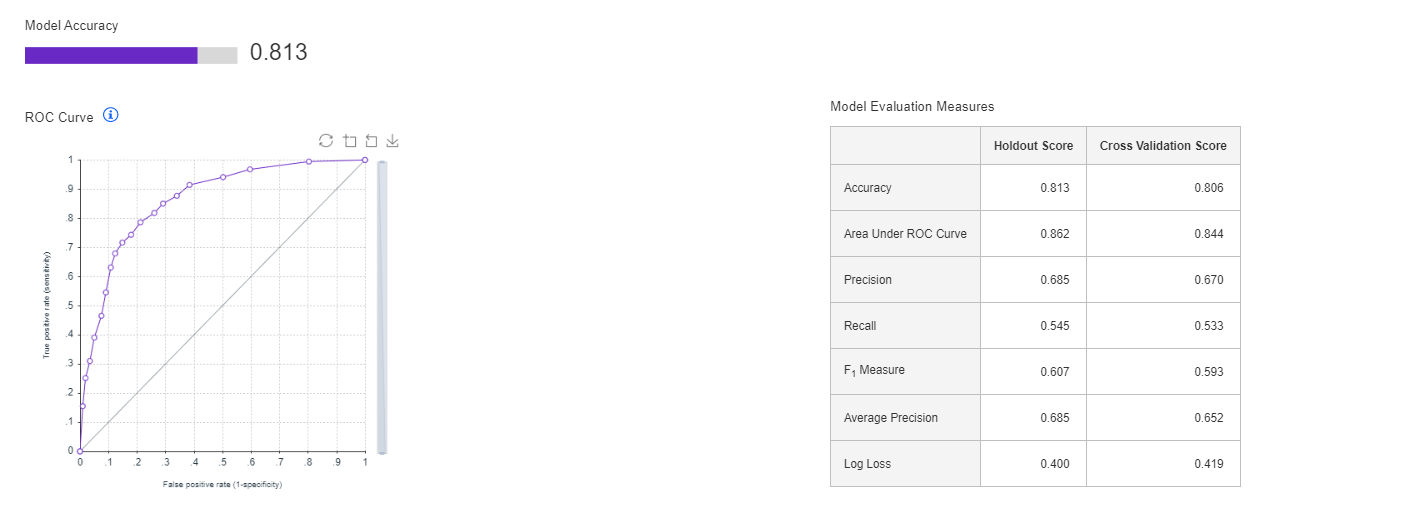
Build the model (pipeline comparison)

Build Flask Application

Save the Model

Data Preparation

1. **RESULT**



The values for accuracy, area under ROC curve, precision, recall, F1 measure,etc are given in the figure above.

1. **ADVANTAGES & DISADVANTAGES**

**Gradient Boosting Trees** build trees one at a time, where each new tree helps to correct errors made by previously trained tree. With each tree added, the model becomes even more expressive. There are typically three parameters - number of trees, depth of trees and learning rate, and the each tree built is generally shallow.

GBDT training generally takes longer because of the fact that trees are built sequentially. However benchmark results have shown GBDT are better learners than Random Forests.

Although it may seem GBDTs are better than random forests, GBDTs are prone to overfitting, however there are strategies to overcome same and build more generalized trees using a combination of parameters like learning rate (shrinkage) and depth of tree. Generally the two parameters are kept on the lower side to allow for slow learning and better generalization.

1. **APPLICATIONS**

It is stated that the costs of acquiring a new customer is five to ten times greater than that of retaining an existing one. As a result of these and similar facts, enterprises are getting more and more interested in customer retention instead of acquiring new customers. Indeed, firms are concluding that the best core marketing strategy for the future is to retain existing customers and avoid customer churn. Customer churn prediction gives managers and marketers the opportunity to design preventing strategies.

1. **CONCLUSION**

This model can be used to predict customer churn rate which is a huge problem faced by many companies. The gradient boosting classifier gives best result for this dataset.

1. **FUTURE SCOPE**

There is a scope of further improvement of this model . As a future work more enhancements can be done to the present algorithm or new algorithms can be formalized.

1. **BIBILOGRAPHY**
2. <https://www.researchgate.net/publication/232631280_Data_Mining_Applications_in_Customer_Churn_Management>
3. https://www.kaggle.com/bandiatindra/telecom-churn-prediction

**APPENDIX**

1. Source code

|  |  |
| --- | --- |
|  | Import json  import os |
|  | from pip.\_vendor import requests |
|  | from dotenv import load\_dotenv |
|  | from flask import Flask, request, session, render\_template, flash |
|  | from requests.auth import HTTPBasicAuth |
|  | import jinja2 |
|  |  |
|  | app = Flask(\_\_name\_\_) |
|  |  |
|  | app.config.update(dict( |
|  | DEBUG=True, |
|  | SECRET\_KEY=os.environ.get('SECRET\_KEY', 'development key') |
|  | )) |
|  |  |
|  | strings = { |
|  | "gender": ['Female', 'Male'], |
|  | "Partner": ['Yes', 'No'], |
|  | "Dependents": ['No', 'Yes'], |
|  | "PhoneService": ['No', 'Yes'], |
|  | "MultipleLines": ['No phone service', 'No', 'Yes'], |
|  | "InternetService": ['DSL', 'Fiber optic', 'No'], |
|  | "OnlineSecurity": ['No', 'Yes', 'No internet service'], |
|  | "OnlineBackup": ['Yes', 'No', 'No internet service'], |
|  | "DeviceProtection": ['No', 'Yes', 'No internet service'], |
|  | "TechSupport": ['No', 'Yes', 'No internet service'], |
|  | "StreamingTV": ['No', 'Yes', 'No internet service'], |
|  | "StreamingMovies": ['No', 'Yes', 'No internet service'], |
|  | "Contract": ['Month-to-month', 'One year', 'Two year'], |
|  | "PaperlessBilling": ['Yes', 'No'], |
|  | "PaymentMethod": ['Electronic check', |
|  | 'Mailed check', |
|  | 'Bank transfer (automatic)', |
|  | 'Credit card (automatic)'] |
|  | } |
|  |  |
|  | # min, max, default value |
|  | floats = { |
|  | "MonthlyCharges": [0, 1000, 100], |
|  | "TotalCharges": [0, 50000, 1000] |
|  | } |
|  |  |
|  | # min, max, default value |
|  | ints = { |
|  | "SeniorCitizen": [0, 1, 0], |
|  | "tenure": [0, 100, 2], |
|  | } |
|  |  |
|  | labels = ["No Churn", "Churn"] |
|  |  |
|  |  |
|  | def generate\_input\_lines(): |
|  | result = f'<table>' |
|  |  |
|  | counter = 0 |
|  | for k in floats.keys(): |
|  | minn, maxx, vall = floats[k] |
|  | if (counter % 2 == 0): |
|  | result += f'<tr>' |
|  | result += f'<td>{k}' |
|  | result += f'<input type="number" class="form-control" min="{minn}" max="{maxx}" step="1" name="{k}" id="{k}" value="{vall}" required (this.value)">' |
|  | result += f'</td>' |
|  | if (counter % 2 == 1): |
|  | result += f'</tr>' |
|  | counter = counter + 1 |
|  |  |
|  | counter = 0 |
|  | for k in ints.keys(): |
|  | minn, maxx, vall = ints[k] |
|  | if (counter % 2 == 0): |
|  | result += f'<tr>' |
|  | result += f'<td>{k}' |
|  | result += f'<input type="number" class="form-control" min="{minn}" max="{maxx}" step="1" name="{k}" id="{k}" value="{vall}" required (this.value)">' |
|  | result += f'</td>' |
|  | if (counter % 2 == 1): |
|  | result += f'</tr>' |
|  | counter = counter + 1 |
|  |  |
|  | counter = 0 |
|  | for k in strings.keys(): |
|  | if (counter % 2 == 0): |
|  | result += f'<tr>' |
|  | result += f'<td>{k}' |
|  | result += f'<select class="form-control" name="{k}">' |
|  | for value in strings[k]: |
|  | result += f'<option value="{value}" selected>{value}</option>' |
|  | result += f'</select>' |
|  | result += f'</td>' |
|  | if (counter % 2 == 1): |
|  | result += f'</tr>' |
|  | counter = counter + 1 |
|  |  |
|  | result += f'</table>' |
|  |  |
|  | return result |
|  |  |
|  |  |
|  | app.jinja\_env.globals.update(generate\_input\_lines=generate\_input\_lines) |
|  |  |
|  |  |
|  | def get\_token(): |
|  | auth\_token = "\*\*\*\*" |
|  | auth\_username = "" |
|  | auth\_password ="" |
|  | auth\_url = "" |
|  |  |
|  | if (auth\_token): |
|  | # All three are set. bad bad! |
|  | if (auth\_username and auth\_password): |
|  | raise EnvironmentError('[ENV VARIABLES] please set either "AUTH\_TOKEN" or ("AUTH\_USERNAME", "AUTH\_PASSWORD", and "AUTH\_URL"). Not both.') |
|  | # Only TOKEN is set. good. |
|  | else: |
|  | return auth\_token |
|  | else: |
|  | # Nothing is set. bad! |
|  | if not (auth\_username and auth\_password): |
|  | raise EnvironmentError('[ENV VARIABLES] please set "AUTH\_USERNAME", "AUTH\_PASSWORD", and "AUTH\_URL" as "TOKEN" is not set.') |
|  | # Only USERNAME, PASSWORD are set. good. |
|  | else: |
|  | response\_preauth = requests.get(auth\_url, auth=HTTPBasicAuth(auth\_username, auth\_password), verify=False) |
|  | if response\_preauth.status\_code == 200: |
|  | return json.loads(response\_preauth.text)['accessToken'] |
|  | else: |
|  | raise Exception(f"Authentication returned {response\_preauth}: {response\_preauth.text}") |
|  |  |
|  |  |
|  | class churnForm(): |
|  |  |
|  | @app.route('/', methods=['GET', 'POST']) |
|  | def index(): |
|  |  |
|  | if request.method == 'POST': |
|  | ID = 999 |
|  |  |
|  | session['ID'] = ID |
|  | data = {} |
|  |  |
|  | for k, v in request.form.items(): |
|  | data[k] = v |
|  | session[k] = v |
|  |  |
|  | scoring\_href = '\*\*\*\*' |
|  |  |
|  |  |
|  | if not (scoring\_href): |
|  | raise EnvironmentError('[ENV VARIABLES] Please set "URL".') |
|  |  |
|  | for field in Ints.keys(): |
|  | data[field] = int(data[field]) |
|  | for field in floats.keys(): |
|  | data[field] = float(data[field]) |
|  |  |
|  | input\_data = list(data.keys()) |
|  | input\_values = list(data.values()) |
|  |  |
|  | payload\_scoring = {"input\_data": [ |
|  | {"fields": input\_data, "values": [input\_values]} |
|  | ]} |
|  | print("Payload is: ") |
|  | print(payload\_scoring) |
|  | header\_online = { |
|  | 'Cache-Control': 'no-cache', |
|  | 'Content-Type': 'application/json', |
|  | 'Authorization': 'Bearer ' + get\_token() |
|  | } |
|  | response\_scoring = requests.post( |
|  | scoring\_href, |
|  | verify=False, |
|  | json=payload\_scoring, |
|  | headers=header\_online) |
|  | result = response\_scoring.text |
|  | print("Result is ", result) |
|  | result\_json = json.loads(result) |
|  |  |
|  | result\_keys = result\_json['predictions'][0]['fields'] |
|  | result\_vals = result\_json['predictions'][0]['values'] |
|  |  |
|  | result\_dict = dict(zip(result\_keys, result\_vals[0])) |
|  |  |
|  | churn\_risk = result\_dict["prediction"].lower() |
|  | no\_percent = result\_dict["probability"][0] \* 100 |
|  | yes\_percent = result\_dict["probability"][1] \* 100 |
|  | flash('Percentage of this customer leaving is: %.0f%%' |
|  | % yes\_percent) |
|  | return render\_template( |
|  | 'score.html', |
|  | result=result\_dict, |
|  | churn\_risk=churn\_risk, |
|  | yes\_percent=yes\_percent, |
|  | no\_percent=no\_percent, |
|  | response\_scoring=response\_scoring, |
|  | labels=labels) |
|  |  |
|  | else: |
|  | return render\_template('input.html') |
|  |  |
|  |  |
|  | load\_dotenv(os.path.join(os.path.dirname(\_\_file\_\_), ".env")) |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | app.run() |