

PROJECT PHASE 3

Bhavani Kukunoor - 50560517

Surya Venkata Rohit Moganti - 50560088

Overview of Phase 1:

In project phase one we have done a lot of data cleaning to our original dataset to train the data with various models. We have removed all the null values using various imputation methods. The outliers in the policy premium column are removed using IQR and all other data inconsistencies like datatypes and categorical values were handled perfectly. We have taken all the numerical columns into consideration to train our model.

Overview of Phase 2:

In project phase 2, Initially we have prepared the dataset by resampling it to avoid bias towards on side of the target class and also in order to avoid uncertain values of the features we have used sklearn's standard scaler to scale all the features in on way. After resampling and scaling we have split the data into training and testing data. Our plan of action was to create a machine learning model using various algorithms and train (fit) the model with the training data that is x_train and y_train. Once the model training is completed, we have tested the accuracy of the model using the testing data. We have also done few visualizations in order to better understand the performance of the model. We have considered various evaluation metrics like precision, f1-score, and recall.

Phase 3 Introduction:

In Phase 2 we have created various machine learning models. In Phase 3 we will be working with only a few models and see how they perform when real life data is given to it. Initially, we will be taking the real time input from the end user, the input from the user will be all the different kinds of features that we have considered. Since we are only using numerical values in order to train the model, we are directly taking numerical input from the user, In order to enhance the user experience we have added a info icon which clearly says the significance of the feature column and based on that the end user can ingest the feature values to the web application.

Before deploying the web application all the machine learning models built by us are saved as .pkl files using a python library called pickle. With this we can save a model and retrieve it whenever necessary. Once the user enters all the feature

values, these values are put into a numpy 2D array and then are fed to the model which is already saved.

Code Overview:

We have utilized the python pickle library in order to save all the pred trained models .pkl format and then re utilize them to predict the new input data.

Saving All the implemented Models

```
[157]: import pickle as pik
```

1 . Linear Regression

```
[158]: LinearReg_model_pkg = 'Models/LinearReg.pkl'

# Use 'wb' to write binary file
with open(LinearReg_model_pkg, 'wb') as file:
    pik.dump(LinearReg, file)
```

```
[ ]:
```

2 . Logistioic Regression

```
[159]: LogisticReg_model_pkg = 'Models/LogisticReg.pkl'

# Use 'wb' to write binary file
with open(LogisticReg_model_pkg, 'wb') as file:
    pik.dump(LogisticReg, file)
```

```
[ ]:
```

3 . Naive Bayes Classifier

```
[160]: nb_classifier_model_pkg = 'Models/nb_classifier.pkl'

# Use 'wb' to write binary file
with open(nb_classifier_model_pkg, 'wb') as file:
    pik.dump(nb_classifier, file)
```

```
[ ]:
```

4 . K Nearest Neighbours

```
[161]: knn_model_pkg = 'Models/knn.pkl'

# Use 'wb' to write binary file
with open(knn_model_pkg, 'wb') as file:
    pik.dump(knn, file)
```

```
[ ]:
```

5 . Kmeans Clustering

```
[162]: kmeans_model_pkg = 'Models/kmeans.pkl'

# Use 'wb' to write binary file
with open(kmeans_model_pkg, 'wb') as file:
    pik.dump(kmeans, file)
```

```
[ ]:
```

6. Decision Tree

```
[163]: dt_model_pkg = 'Models/dt.pkl'

# Use 'wb' to write binary file
with open(dt_model_pkg, 'wb') as file:
    pik.dump(dt_classifier, file)
```

7. Neural Network

```
64]: nn_model_pkg = 'Models/nn.pkl'

# Use 'wb' to write binary file
with open(nn_model_pkg, 'wb') as file:
    pik.dump(nn, file)
```

```
[ ]:
```

8. Random Forest

```
65]: random_forest_model_pkg = 'Models/random_forest.pkl'

# Use 'wb' to write binary file
with open(random_forest_model_pkg, 'wb') as file:
    pik.dump(random_forest, file)
```

```
[ ]:
```

After saving the models we create an app.py file where we read all the saved .pkl files and also fetch the user input from the index webpage. Then we process the input data by scaling and converting it into a numpy array. We later feed the input data to the machine learning model and then store the result in an output variable, then we render the result page and transfer the output variable to the result page and then process it to be displayed accordingly.

```
1 from flask import Flask, request, render_template
2 import pickle
3 import numpy as np
4 from imblearn.over_sampling import SMOTE
5 from sklearn.preprocessing import StandardScaler
6
7 app = Flask(__name__)
8
9 # Execute the following commands in the terminal if the error says there is an already existing service in port 5000
10 # sudo lsof -i :5000
11 # sudo kill -9 <PID>
12
13 # List of paths to your machine learning model files
14 model_paths = [
15     ('Decision Tree', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/dt.pkl'),
16     ('K Means', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/kmeans.pkl'),
17     ('KNN', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/knn.pkl'),
18     ('Linear Regression', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/LinearReg.pkl'),
19     ('Logistic', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/LogisticReg.pkl'),
20     ('Naive Bayes', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/nb_classifier.pkl'),
21     ('Neural Network', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/nn.pkl'),
22     ('Random Forest', '/Users/rohithsurya/Documents/Lab/DIC_LAB/SuryaVenkataRohit_BhavaniKiran_phase_2/Phase_3/Models/random_forest.pkl')
23 ]
24 # Loading Machine Learning model from specific path
25 models = {}
26
27 for name, path in model_paths:
28     with open(path, 'rb') as f:
29         models[name] = pickle.load(f)
30
31 @app.route('/')
32 def home():
33     return render_template('index.html')
34
35 @app.route('/predict', methods=['POST'])
36 def predict():
37     # The input data from the website is retrieved here
38     input_data = [float(request.form[f'value{i+1}']) for i in range(20)]
39     #print(input_data)
40     input_array = np.array(input_data)
41     temp_mean = np.mean(input_array)
42     temp_std = np.std(input_array) if np.std(input_array) != 0 else 1.0 # Avoid division by zero; choose a reasonable std
43
```

```
45 # Scaling the data Manually
46 scaled_data = (input_array - temp_mean) / temp_std
47
48 #print(scaled_data)
49
50 # Feeding the input data to saved models
51 results = {}
52 for name, model in models.items():
53     # Selected Model from UI
54     print(request.form['value21'])
55     if name == request.form['value21']:
56         prediction = model.predict(scaled_data)
57         accuracy = model.score(scaled_data, prediction)
58
59         results[name] = (prediction[0], accuracy)
60
61 # Post the Results to the result website
62 return render_template('result.html', results=results)
63
64 if __name__ == "__main__":
65     app.run(debug=True)
66
```

Models Used:

We have used all the models which we have developed in project phase 2, We have given an option to the user on which model they want to test with their input data. In the output from the result page, we will be displayed what Machine Learning model did the user select and with what accuracy did the Model predict the given input data.

How to Run the Code:

Initially execute the last part in the .ipynb file where the phase two models are saved in .pkl format using python pickle library.

Then Navigate to the app directory and in the command prompt hit the command `python3 app.py` this executes the app.py file and then renders together the app.py and index.html, result.html file.

The app.py code then displays the hyperlink to the localhost port server where the web application is hosted. Copy the link from the terminal and paste it in the web browser of choice.

Recommendations on Model Based on Analysis:

Discover High-Value customers: Utilizing predictive scoring, our software can assist clients in choosing between high- and low-value prospective potential customers. This enables users to assign resources to leads that have a higher chance of changing, hence maximizing return on investment.

Market Dynamics Visualization: Our product can provide insights into market trends, such as increasing or decreasing demand for specific insurance product types among specific demographics by gathering lead prediction data over time.

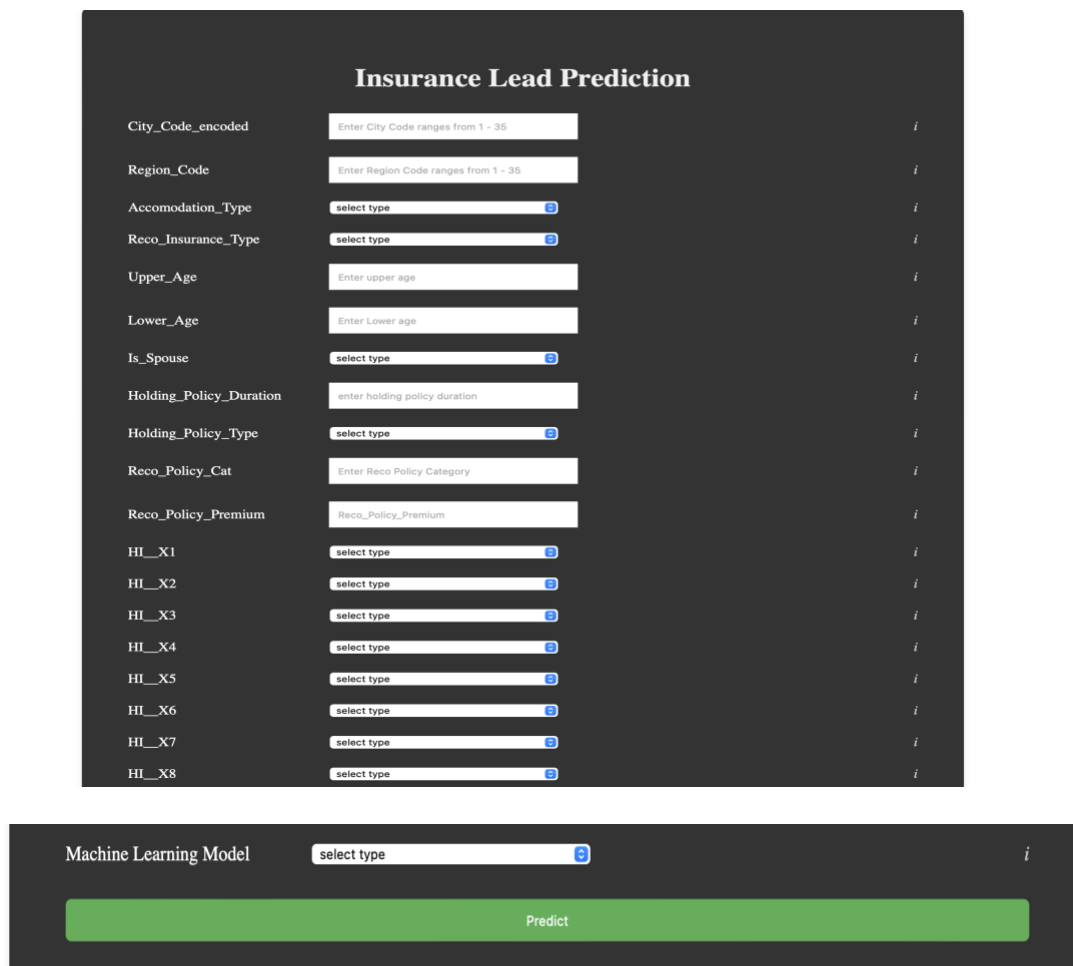
Ideas for Project extension

Real-time Data Feeding: Make the product capable of collecting real-time data so that lead projections can be refreshed and improved continuously based on the latest information available.

Connectivity to CRM Systems: Develop extensions that allow the lead prediction tool to be easily integrated with the customer relationship management (CRM) programs that insurance companies now use.

Graphical User Interface:

Input data:



The image displays a web-based graphical user interface titled "Insurance Lead Prediction". It features a dark-themed layout with a list of input fields on the left and a "Predict" button at the bottom. The input fields are organized into two main sections. The first section contains fields for "City_Code_encoded", "Region_Code", "Accommodation_Type", "Reco_Insurance_Type", "Upper_Age", "Lower_Age", "Is_Spouse", "Holding_Policy_Duration", "Holding_Policy_Type", "Reco_Policy_Cat", and "Reco_Policy_Premium". The second section contains eight fields labeled "HI__X1" through "HI__X8". Each field has a specific input type: text boxes for codes, ages, and duration; dropdown menus for types and categories; and a text box for premium. A "Machine Learning Model" dropdown is located at the bottom left, and a large green "Predict" button is centered at the bottom. The interface is clean and modern, with a focus on user input.

Field Name	Input Type
City_Code_encoded	Text (Range: 1 - 35)
Region_Code	Text (Range: 1 - 35)
Accommodation_Type	Dropdown (select type)
Reco_Insurance_Type	Dropdown (select type)
Upper_Age	Text (Enter upper age)
Lower_Age	Text (Enter Lower age)
Is_Spouse	Dropdown (select type)
Holding_Policy_Duration	Text (enter holding policy duration)
Holding_Policy_Type	Dropdown (select type)
Reco_Policy_Cat	Text (Enter Reco Policy Category)
Reco_Policy_Premium	Text (Reco_Policy_Premium)
HI__X1	Dropdown (select type)
HI__X2	Dropdown (select type)
HI__X3	Dropdown (select type)
HI__X4	Dropdown (select type)
HI__X5	Dropdown (select type)
HI__X6	Dropdown (select type)
HI__X7	Dropdown (select type)
HI__X8	Dropdown (select type)
Machine Learning Model	Dropdown (select type)
Predict	Button

In the above text fields on the website, we have to enter all the details about the customer and this data is the feature data. Once the data is entered, we need to select the type of Machine Learning model that we need to test, we have various options for the machine learning models. The data is then stores in a python dictionary along with the name of the model. Then this input data is transformed

into an np 2D array. This input is then scaled using a temporary scaling process to deal within consistent data. The encoded columns are given as it is, as the importance of these feature columns are very less. This np array is then given to the already saved model and the output is obtained. This output is then rendered to the result.html page which is programmed to display the output.

Output Data:

Results

Decision Tree: Prediction: 1, Accuracy: 1.0%

This is an insurance lead.

[Try Again](#)

Results

K Means: Prediction: 0, Accuracy: -17.56%

This is not an insurance lead.

[Try Again](#)

The output which we get after doing the predict operation with the input data to the model, that output is then rendered to the result.html page. Here we take the output variable which can be 0 or 1 and then we process it. If we get result as 0 then the website displays that the input data provided is not an insurance lead. If we get result as 1 then the website displays that the input data provided is an insurance lead.

Conclusion :

As this phase of the insurance lead prediction project comes to an end, it's clear that the creation of a machine learning-powered application has enhanced the efficiency and accuracy of finding and assessing potential insurance leads as well as opened up possibilities for revolutionary changes in the way insurance companies interact with their customers.

References :

Flask API : <https://flask.palletsprojects.com/en/latest/api/>

Pickle : <https://docs.python.org/3/library/pickle.html>