Names: Eugene Chang, Sanho Lee

NYU IDs: N174044284, N15250101

Course Section Number: CSCI-GA.2433-001 - Fall 2023

Date: 12/7/2023

**EDA Project Part 3**

Goals for Part 3 Documentation:

* Filled azure sql server with dataset
  + Going to train AI model on this dataset
  + What parameters?
    - Driver age, (years of driving experience), car mileage, driving experience, gender, etc.
* Exploring supervised vs unsupervised learning
* Goal website:
  + User inputs parameters above to get a quote (predicted insurance cost)
  + User receives predicted insurance quote
  + Web frameworks
    - Express (Javscript)
    - Python (**Django**, flask)

Our User Case Diagram

[User Logs In]

|

|--> [View Policy Details]

| |

| |--> [Update Personal Info] ----> [Changes Saved]

| |

| |--> [Update Vehicle Info] -----> [Changes Saved]

|

|--> [Renew Policy]

| |

| |--> [Make Payment] ----> [Policy Renewed]

|

|--> [File a Claim]

|

|--> [Enter Claim Details] ----> [Claim Submitted]

—--> [Know your Estimated Cost (Get a quote today!)]

|

| ⇒ [Enter their information] —--> Server uses machine learning algorithm on this information --> [Show them their estimated Cost]

**Changes that we made:**

We added a new Table Called Driving History to apply this table into machine learning algorithms to predict our estimated insurance cost.

DRIVING\_HISTORY

* CustomerSSN
* Traffic Violations
* Accidents
* DrivingExperience

We used **Linear Regression** (Supervised) Algorithm to predict an estimated insurance cost(Maximum Coverage) because insurance fee is a continued variable.

However, when we ran the code, we ran into a problem of relatively high MSE. The reason was we used dummy data to fill our datasets for our TABLE. We used a relevant factor of

CUSTOMER:

* AGE
* CustomerGender

VEHICLE:

* Mileage

DRIVING\_HISTORY

* Traffic Violations
* Accidents
* DrivingExperience

Since these were all filled with dummy data, there was no relevance between these factors and the target variable ‘MaximumCoverage’.

**So our next goal is** to modify our dataset manually to make a relevance between these factors and ‘MaximumCoverage’ Such as setting coverage fees relatively lower to customers who have high accidents and higher Traffic Violations, etc.

////////////////////////

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from datetime import datetime

import matplotlib.pyplot as plt

df\_customer = pd.read\_csv('customer.csv')

df\_vehicle = pd.read\_csv('vehicle.csv')

df\_driving\_history = pd.read\_csv('driving\_history.csv')

df\_contract = pd.read\_csv('contract.csv')

# Join dataframes

df = pd.merge(df\_customer, df\_vehicle, on='CustomerSSN', how='inner')

df = pd.merge(df, df\_driving\_history, on='CustomerSSN', how='inner')

df = pd.merge(df, df\_contract, on='CustomerSSN', how='inner')

# Convert DOB to Age

df['CustomerDOB'] = pd.to\_datetime(df['CustomerDOB'])

df['Age'] = df['CustomerDOB'].apply(lambda x: datetime.now().year - x.year)

# Selecting relevant features

features = ['Age', 'CustomerGender', 'Mileage', 'VehicleType', 'TrafficViolations', 'Accidents', 'DrivingExperience']

X = df[features]

# Assuming 'MaximumCoverage' is your target variable

y = df['MaximumCoverage']

# Preprocessing

numeric\_features = ['Age', 'Mileage', 'TrafficViolations', 'Accidents', 'DrivingExperience']

categorical\_features = ['CustomerGender', 'VehicleType']

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numeric\_features),

('cat', OneHotEncoder(), categorical\_features)])

# Create a pipeline

model = Pipeline(steps=[('preprocessor', preprocessor),

('regressor', LinearRegression())])

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

# Predict and evaluate

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

for actual, predicted in zip(y\_test[:10], y\_pred[:10]):

print(f"Actual: {actual}, Predicted: {predicted}")

plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual Coverage Fees")

plt.ylabel("Predicted Coverage Fees")

plt.title("Actual vs Predicted Coverage Fees")

plt.show()

///////////////////////////////////

Supervised Learning Idea

* We can create sample profiles of drivers with different preset profiles
  + Profile: Age, Gender, Milage, Traffic Violations, Accidents, Driving Experience
  + Manually set a monthly insurance cost for this profile

Overview:

\*\*Splitting the Data\*\*: Divide your dataset into a training set and a test set. This allows you to evaluate how well your model is likely to perform on unseen data.

\*\*Model Training\*\*: Choose a supervised learning algorithm and train it on your training set. This involves showing the algorithm the input features along with the correct output (the insurance premium), so it can learn to map the inputs to the correct output.

\*\*Model Evaluation\*\*: Use your test set to evaluate the performance of your model. This involves using the model to predict the insurance premiums for the test set, and then comparing these predictions to the actual premiums.

* Not sure if we have to do this

\*\*Prediction\*\*: Once your model is trained and evaluated, you can use it to predict the insurance premium for a new user. This involves taking the user's input values, running them through your model, and outputting the predicted insurance premium.!

**Goals for Part 4:**

* Solidify machine learning algorithm and learning method
  + Supervised Learning
    - We have to create sample data manually that is “accurate” to train on
  + Unsupervised
    - Really hard since dataset is not realistic
* Create a basic website based on the User Case Diagram we mentioned above
  + Takes in user input as a form
  + Sends request to the server that runs the Machine Learning algorithm
    - Server is connected to Azure SQL server
  + Returns a response with a predicted insurance cost for this user
* Publish everything on GitHub at the end
  + Gif demo of the app
  + Instructions on how to run server and website on local host