**6006 CEM Machine learning coursework. Analyzing car insurance claims dataset**

**Coventry GitHub Repository URL** or **Coventry OneDrive URL** (mandatory):

https://livecoventryac-my.sharepoint.com/:f:/g/personal/zimuse\_uni\_coventry\_ac\_uk/EoNAKQpz6UBGnj9QkqW837cBehLYZla3bNEE5YvESJEndQ?e=9s34eR

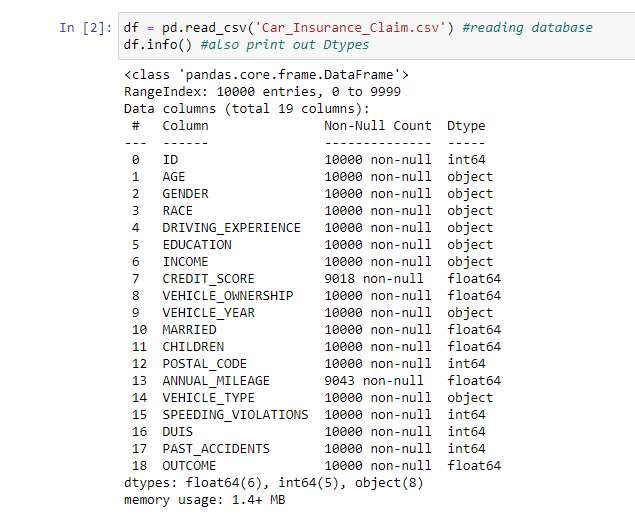
|  |
| --- |
| **Academic Report** |

# Introduction

Nowadays car insurance is an actual topic to talk about, different countries have different models how car insurance is sold, whether you insure a vehicle or a driver driving specific vehicle. I have retrieved a dataset from Kaggle with car insurance claims data that contains 10000 rows and 19 columns (Roy, 2021). The problem itself is to decide from data whether a person is likely to make an insurance claim. To solve this problem I build a machine learning approach using logistic regression, K neighbors, neural network learning models. They are all capable of making predictions.

# Pre-processing data

To begin with I check dtypes



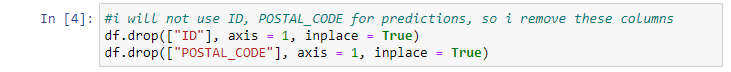
Also check how dataset looks like:

Graphical user interface, text, application, email

Description automatically generated

Perhaps there is string data (under object dtypes) which is not suitable for predictions, it will have to be converted to int32, most suitable and simple solution for that is to use a label encoder.

Also, upfront I remove ID and POSTAL\_CODE columns as they are not relevant to the prediction models:



I need to check if there are any null values:

Table

Description automatically generated

I could just drop these columns from the dataset, but I will not as I would lose 1939 rows of data, instead I can fill them in with mean values:

Graphical user interface

Description automatically generated

Why fill in with mean values? There are plenty of rows with data and what I am filling in is credit score and mileage, credit score is hard to gain and is not a volatile variable, miles are for an average customer. It would not be possible to fill in columns with mean to married or children columns etc..

As I said, I will have to convert the object columns with help of label encoder:

Text

Description automatically generated

After conversion dtypes look like that:

Table

Description automatically generated with medium confidence

Perhaps it can be compared how dataset looked like and how it looks now:

Graphical user interface, text, application, chat or text message

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Afterwards I split the dataset and prepare it for learning algorithms, 60% is used for training with random state of 1:

Text

Description automatically generated

Now it is ready to apply various algorithms.

# Applying different algorithms and methods to build learning models

Logistic regression:

Text

Description automatically generated

KN classifier:

Text

Description automatically generated

Neural network:

Text, letter

Description automatically generated

All of them have similar accuracy score, I chose to apply these models as they are relatively easy to apply, and they are suitable for that dataset assuming that 80%+ accuracy is great. I chose to use logistic regression because it is valuable to predict a likelihood of an event (What Is Logistic Regression? Learn When to Use It, 2021). KNN uses proximity to make classifications or predictions about the grouping of an individual data point (What Is the K-nearest Neighbors Algorithm? | IBM, n.d.). Neural network is for deep learning, creates an artificial brain that uses interconnected nodes or neurons in a layered structure. (What Is a Neural Network? AI and ML Guide - AWS, n.d.).

# Making appropriate adjustments to improve the model performance

As pre processing was done, after training the model the scaling was done, it increased the model accuracy by around 3%. And I decided to do 60% data for training, it gives the best result, and changing that value the model accuracy differs by around 0.001-0.005%.

# Evaluating models

To finalise, there were 3 learning models used for this work, logistic regression, K nearest neighbors, neural network.

Text

Description automatically generated

From just the accuracy it can be said that logistic regression is performing the best out of all 3.

Moving on to confusion matrixes for each model, to read them please refer to this example:

|  |  |
| --- | --- |
| True positive | False positive |
| False negative | True negative |

Text

Description automatically generated

The confusion matrix visualises what is really happening meaning of guesses whether they were true positive, true negative, false positive, false negative, it refers to the accuracy score.

Classification report for each model was also done. It shows the quality of the predictions from a classification algorithm. (Understanding the Classification Report Through Sklearn – Muthukrishnan, 2018)

A picture containing text, receipt

Description automatically generated

Perhaps cross validation was done too:

Graphical user interface, text, application

Description automatically generated

Cross validation does training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. (Cross-Validation - Amazon Machine Learning, n.d.)

Overall the best model to use in this case is logistic regression, even the accuracy differences by little to none, it is best to chose the highest number in this scenario.

# Comparing the approaches and results of other existing pieces of work on the same problem

There are plenty of existing pieces of work with more data visualisation, one of them using xgBoost, catBoost (picked best performing ones) models, giving accuracies of 85%, 86%, (VELU, 2022), but that specific work does not use logistic regression and neural network models.

There is also another users work (SHILAWANE, 2022), it contains a logistic regression model and there is classification report with f1-score averaging 0.85 and mine averages to 0.828. f1 score is a weighted harmonic mean of precision and recall (Classification Report — Yellowbrick v1.5 Documentation, n.d.). Meaning of the higher the f1 score is the better the model is itself.

[**word count**: 768]

|  |
| --- |
| **Bibliography** |

*Classification Report — Yellowbrick v1.5 documentation*. (n.d.). https://www.scikit-yb.org/en/latest/api/classifier/classification\_report.html

*Cross-Validation - Amazon Machine Learning*. (n.d.). https://docs.aws.amazon.com/machine-learning/latest/dg/cross-validation.html

*Free Citation Generator | APA, MLA, Harvard*. (2022, November 8). Scribbr. https://www.scribbr.com/citation/generator/

Roy, S. (2021, July 5). *Car Insurance Data*. Kaggle. https://www.kaggle.com/datasets/sagnik1511/car-insurance-data

*scikit-learn: machine learning in Python — scikit-learn 1.1.3 documentation*. (n.d.). https://scikit-learn.org/stable/

*seaborn.heatmap — seaborn 0.12.1 documentation*. (n.d.). https://seaborn.pydata.org/generated/seaborn.heatmap.html?highlight=heatmap

SHILAWANE, Y. (2022, August 14). *Car\_Insurance\_Data*. Kaggle. https://www.kaggle.com/code/yugandharshilawane/car-insurance-data

*Source code beautifier / syntax highlighter – convert code snippets to HTML « hilite.me*. (n.d.). http://hilite.me/

*Understanding the Classification report through sklearn – Muthukrishnan*. (2018, July 7). https://muthu.co/understanding-the-classification-report-in-sklearn/

VELU, A. (2022, October 6). *Car Insurance Predication*. Kaggle. https://www.kaggle.com/code/aadarshvelu/car-insurance-predication

*What is a Neural Network? AI and ML Guide - AWS*. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/neural-network/

*What Is Logistic Regression? Learn When to Use It*. (2021, July 29). https://learn.g2.com/logistic-regression

*What is the k-nearest neighbors algorithm? | IBM*. (n.d.). https://www.ibm.com/topics/knn

|  |
| --- |
| **Appendix A** |

< A suggested checklist for you, for full details please refer to the coursework brief >

1. The following naming convention is used for the Coventry GitHub Repository and Coventry OneDrive

StudentID-Initials-s1

For example, a student Liz Truss whose student ID is 12345678 would name their repository or shared folder as **12345678-LT-s1**

Failing to follow the naming convention may delay the release of marks and feedback for your coursework.

1. **Coventry** GitHub Repository URL **or** **Coventry** OneDrive URL: added to the top of this report
   1. Coventry GitHub Repository includes:

* URL of the selected dataset(s) included in README
* The original selected dataset(s)
* Source-code (.ipynb)
* Demonstration video (.mp4)
  1. Coventry OneDrive folder includes:
* URL of the selected dataset(s) included in a separated text file
* The original selected dataset(s)
* Source-code (.ipynb)
* Demonstration video (.mp4)

1. Source-code added **as text** in Appendix B (below)
2. Submission in the form of a **Word** document. *\*\*Other format is not accepted.*

|  |
| --- |
| **Appendix B** |

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** datetime **import** datetime

**from** sklearn.preprocessing **import** LabelEncoder

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

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** accuracy\_score**,** confusion\_matrix**,** classification\_report

**from** sklearn.svm **import** SVC

**from** sklearn.model\_selection **import** cross\_val\_score

**from** sklearn.neural\_network **import** MLPClassifier

**import** warnings

warnings**.**filterwarnings**(**'ignore'**)**

df **=** pd**.**read\_csv**(**'Car\_Insurance\_Claim.csv'**)** *#reading database*

df**.**info**()** *#also print out Dtypes*

df**.**head**()** *#show the abstract database*

*#i will not use ID, POSTAL\_CODE for predictions, so i remove these columns*

df**.**drop**([**"ID"**],** axis **=** **1,** inplace **=** True**)**

df**.**drop**([**"POSTAL\_CODE"**],** axis **=** **1,** inplace **=** True**)**

df**.**head**()** *#check*

df**.**isnull**().**sum**().**sum**()** *# count how many null values there are*

df**.**isnull**().**sum**()** *# credit\_score and annual\_mileage has null values*

np**.**mean**(**df**[**"CREDIT\_SCORE"**])** *#check mean value for credit score*

np**.**mean**(**df**[**"ANNUAL\_MILEAGE"**])** *#check mean value for annual mileage*

df**[**"CREDIT\_SCORE"**]** **=** df**[**"CREDIT\_SCORE"**].**fillna**(**np**.**mean**(**df**[**"CREDIT\_SCORE"**]))**

df**[**"ANNUAL\_MILEAGE"**]** **=** df**[**"ANNUAL\_MILEAGE"**].**fillna**(**int**(**np**.**mean**(**df**[**"ANNUAL\_MILEAGE"**])))**

*#fill null values with mean*

df**.**isnull**().**sum**()** *#check if all null values were filled in*

*#this block is to transform the values to be suitable for training purposes*

df**[**"GENDER"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"GENDER"**])**

df**[**"VEHICLE\_TYPE"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"VEHICLE\_TYPE"**])**

df**[**"VEHICLE\_YEAR"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"VEHICLE\_YEAR"**])**

df**[**"AGE"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"AGE"**])**

df**[**"DRIVING\_EXPERIENCE"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"DRIVING\_EXPERIENCE"**])**

df**[**"EDUCATION"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"EDUCATION"**])**

df**[**"INCOME"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"INCOME"**])**

df**[**"RACE"**]** **=** LabelEncoder**().**fit\_transform**(**df**[**"RACE"**])**

df**.**info**()** *#check dtypes*

df**.**head**()** *#processed dataset*

x\_train**,** x\_test**,** y\_train**,** y\_test **=** train\_test\_split**(**df**.**iloc**[** **:** **,** **:** **-1],** df**.**iloc**[** **:** **,** **-1],** train\_size **=** **.6,** random\_state **=** **1)**

*#prepare dataset for training*

*#included scaler as it gives better accuracy*

x\_train **=** StandardScaler**().**fit\_transform**(**x\_train**)**

x\_test **=** StandardScaler**().**fit\_transform**(**x\_test**)**

*#logistic regression*

model **=** LogisticRegression**(**max\_iter**=2000)**

model**.**fit**(**x\_train**,**y\_train**)**

predicted**=**model**.**predict**(**x\_test**)**

**print(**"accuracy of logistic regression is: "**,**accuracy\_score**(**y\_test**,** predicted**))**

a1**=**accuracy\_score**(**y\_test**,** predicted**)**

*#confusion matrix for logistic regression*

y\_pred **=** model**.**predict**(**x\_test**)**

cf\_matrix **=** confusion\_matrix**(**y\_test**,** y\_pred**)**

**print(**'Confusion matrix\n\n'**,** cf\_matrix**)**

sns**.**heatmap**(**cf\_matrix**,** annot**=**True**,** fmt**=**'d'**,** cmap**=**'Greens'**)**

a2**=**cf\_matrix

*#classification report for logistic regression model*

**print(**classification\_report**(**y\_test**,** y\_pred**))**

a3**=**classification\_report**(**y\_test**,** y\_pred**)**

*#cross validation for logistic regression model*

score **=** cross\_val\_score**(**model**,** x\_train**,** y\_train**,** cv **=** **5,** scoring**=**'accuracy'**)**

**print(**'Cross-validation scores:{}'**.**format**(**score**))**

**print(**'Average cross-validation score: {:.3f}'**.**format**(**score**.**mean**()))**

a4**=**score

*#knn classifier*

model **=** KNeighborsClassifier**(**n\_neighbors **=** **5)**

model**.**fit**(**x\_train**,** y\_train**)**

predicted**=**model**.**predict**(**x\_test**)**

a5**=**accuracy\_score**(**y\_test**,** predicted**)**

**print(**"Accuracy of knn is: "**,** accuracy\_score**(**y\_test**,** predicted**))**

*#confusion matrix for knn*

y\_pred **=** model**.**predict**(**x\_test**)**

cf\_matrix **=** confusion\_matrix**(**y\_test**,** y\_pred**)**

**print(**'Confusion matrix\n\n'**,** cf\_matrix**)**

sns**.**heatmap**(**cf\_matrix**,** annot**=**True**,** fmt**=**'d'**,** cmap**=**'Greens'**)**

a6**=**cf\_matrix

*#classification report for knn model*

**print(**classification\_report**(**y\_test**,** y\_pred**))**

a7**=**classification\_report**(**y\_test**,** y\_pred**)**

*#cross validation for knn model*

score **=** cross\_val\_score**(**model**,** x\_train**,** y\_train**,** cv **=** **5,** scoring**=**'accuracy'**)**

**print(**'Cross-validation scores:{}'**.**format**(**score**))**

**print(**'Average cross-validation score: {:.3f}'**.**format**(**score**.**mean**()))**

a8**=**score

*#neural network*

model **=** MLPClassifier**(**random\_state**=0)**

model**.**fit**(**x\_train**,** y\_train**)**

score**=** model**.**score**(**x\_test**,** y\_test**)**

a9**=**score

**print(**'Testing score: {:.3f}'**.**format**(**score**))**

score **=** model**.**score**(**x\_train**,** y\_train**)**

a10**=**score

**print(**"Training score: {:.3f}"**.**format**(**score**))**

*#confusion matrix for neural network model*

y\_pred **=** model**.**predict**(**x\_test**)**

cf\_matrix **=** confusion\_matrix**(**y\_test**,** y\_pred**)**

**print(**'Confusion matrix\n\n'**,** cf\_matrix**)**

sns**.**heatmap**(**cf\_matrix**,** annot**=**True**,** fmt**=**'d'**,** cmap**=**'Greens'**)**

a11**=**cf\_matrix

*#classification report for neural network model*

**print(**classification\_report**(**y\_test**,** y\_pred**))**

a12**=**classification\_report**(**y\_test**,** y\_pred**)**

*#cross validation for neural network model*

score **=** cross\_val\_score**(**model**,** x\_train**,** y\_train**,** cv **=** **5,** scoring**=**'accuracy'**)**

**print(**'Cross-validation scores:{}'**.**format**(**score**))**

**print(**'Average cross-validation score: {:.3f}'**.**format**(**score**.**mean**()))**

a13**=**score

*#simplified view for evaluation*

**print(**"---Evaluation---"**)**

**print(**"Accuracy:"**)**

**print(**"Accuracy of logistic regression is: "**,**a1**)**

**print(**"Accuracy of knn is: "**,** a5**)**

**print(**'Neural network Testing score: {:.3f}'**.**format**(**a9**))**

**print(**"Neural network Training score: {:.3f}"**.**format**(**a10**))**

**print(**"----------------"**)**

**print(**"Confusion matrix"**)**

**print(**"Logistic regression"**)**

**print(**a2**)**

**print(**"KNN classifier"**)**

**print(**a6**)**

**print(**"Neural network"**)**

**print(**a11**)**

**print(**"----------------"**)**

**print(**"Classification report"**)**

**print(**"Logistic regression"**)**

**print(**a3**)**

**print(**"KNN classifier"**)**

**print(**a7**)**

**print(**"Neural network"**)**

**print(**a12**)**

**print(**"----------------"**)**

**print(**"Cross validation"**)**

**print(**"Logistic regression"**)**

**print(**a4**)**

**print(**a4**.**mean**())**

**print(**"KNN classifier"**)**

**print(**a8**)**

**print(**a8**.**mean**())**

**print(**"Neural network"**)**

**print(**a13**)**

**print(**a13**.**mean**())**

**print(**"----------------"**)**