Software Development for Artificial Intelligence

CSCE 5214, Fall 2022

UNT™

UNIVERSITY OF NORTH TEXAS

Project Title: Health Insurance Premium Prediction using AWS Cloud

Team Members

- **Arun Sai Kumar Gutala**
- **Sindhuja Chepuri**
- **Hari Vamsi Krishna Samayamantri**

GitHub Wiki URL - https://github.com/akgutala/SDAI_Project-HIPP_System/wiki

GitHub Repository URL - https://github.com/akgutala/SDAI_Project-HIPP_System

1. Motivation

Prediction systems have always been the core application of Machine Learning (M.L.) and in this project we are trying to visualize and predict the health insurance premium which is calculated over several factors like age, sex, geographic locations. Most of the models currently developed are based on Ridge Regression and we would like to improve this by using Multiple Linear Regression and make use of regression metrics like Root-mean-square deviation(RMSE), Mean squared error (MSE) and if time permits would like to include some other metrics as well. This predictor will make use of AWS cloud services like AWS Sagemaker, AWS S3, etc.

2. Significance

This project can be used for any domain that might require a prediction which can be trained on the given dataset which gives promising predictions. The health insurance premium prediction will ensure that the people will stay healthy in certain parameters so that the premium always stays low and also know how their premium is going to look in the coming future with increased parameters like BMI, region. Foreseeing things has always been a benefit all the time. This will also help in fast-settlement of insurance claims by analyzing factors based on business policies.

3. Objectives

The main objective of our project is to know about the price of the medical insurance premium incurred by individuals based on different parameters like age, gender, BMI, smoking, geo-location by training an artificial intelligence regression based model in AWS Sagemaker. Creating new training and testing datasets and deploying the model with endpoints.

4. Features

- Evaluating the dataset and training the model with linear regression first and improving with multiple linear regression.
- Visualization of the dataset and perform exploratory data analysis.
- Easily accessible end points to make use of the predicted model.

Related Work (Background)

- Introduction to Artificial Intelligence and Machine Learning

AI refers to the algorithms and processes that mimic cognitive functions like learning and problem solving. Machine learning and Deep Learning are subsets of AI.

Some applications of AI which are used in our day-to-day life include personal assistants that understand spoken language, web search engines, recommendation engines used by Spotify and Netflix, self-driving vehicles.To begin with, there are 4 types of AI. They include reactive machines, limited theory, theory of mind and self-awareness. Reactive Machines will be able to perform basic operations based on some input. They do not store inputs.Limited Memory systems store the input data, data about actions, decisions. This data can be stored, analyzed and improved over time. Examples include chatbots, self-driving vehicles, voice assistants. In Theory of mind, AI would understand human thoughts and emotions and start interacting in a meaningful way. Self-awareness is where AI has human level consciousness.

Machine Learning falls within the Limited Memory category of AI. It's a subset of AI. Mainly there are three types of learning algorithms. They are supervised learning, unsupervised learning and reinforcement learning. Supervised Learning is when AI is supervised all along the learning process. We provide a labeled dataset and a model can learn from the input data and provide the result. It is mainly used in classification and regression problems. Unsupervised learning discovers the underlying patterns. It deals with unlabeled data. It's mainly used in clustering problems.

Reinforcement Learning algorithms learn on their own on how to react to an environment. It's not particularly supervised or unsupervised learning. It follows a trial and error method. In this an agent tries to manipulate the environment. On failure, the agent will not get any reward. But on success it gets success in the environment. In this way the agent learns how the environment works.

- Regression

Regression is a part of supervised learning technique which is used to model the relationship between independent and dependent variables. Mainly it helps us understand how the dependent variable changes when there are corresponding changes in the independent variable. It is mainly used for forecasting, weather predictions, time series modeling, prediction etc.

In regression we plot a graph between the different variables and based on this graph we can make predictions. We plot a line or curve which passes through all the data points in a way that the distance between the regression line and data points is minimum. There are many types of regression. Some of which are Linear Regression, Random Forest Regression, Logistic Regression, Decision Tree Regression, Polynomial Regression, Lasso Regression , Support Vector Regression, Ridge Regression

If our algorithm works well with training data but not with test data, we refer to such problems as Overfitting. If algorithm doesn't perform well with either test or training dataset its called Underfitting

- SaaS:

Software as a service (or SaaS) , sometimes called Web-based software is a way of developing applications over the internet. SaaS applications run on SaaS providers servers. Providers manage security, performance, access to the application, availability. Advantages of Saas are : Scalability, Access from anywhere, low setup costs, low infrastructure cost and mainly security at its highest level.

- Sagemaker:

Amazon Sagemaker is a managed service in AWS Cloud. Sage Maker aims to ease the processing of deploying the ML models. It provides tools required to build, train,deploy ML models. ML requires dedicated hardware and workflow tools to process datasets. Two major steps involved are training and inferencing. Based on repeated pattern recognition machines learn to behave in a certain way.

SageMaker handles ML in 3 steps:

★ Preparation
★ Training
★ Deployment

Developers can login to SageMaker console , pull data sets from S3 instances. It provides a variety of algorithms like linear regression, image classification. Next we train our data.Here we transform the data to enable feature engineering. When the model is deployment ready, a service automatically scales and operates the cloud infrastructure. Along with these functions sagemaker applies security patches, performs health checks, and establishes HTTPS endpoints.

- **ANN**:

Artificial Neural Networks are gaining a huge popularity. These machine learning algorithms have gained this popularity because of the increased computation power.

They are modeled after the human brain. ANN learns from the data and provides responses in the form of classifications and predictions.

ANNs contain three main layers :Input Layer, Hidden Layer, Output Layer. There are multiple parameters and hyperparameters in a neural networks.Some of the input parameters are biases, weights, batch size, learning rate. Using a transfer function weighted sum of inputs and bias are calculated. Then the activation function obtains the final result. Some of the popular functions are Sigmoid, Softmax, RELU,tanh etc.

- **There are 2 types of ANN's:**

1) Feed Forward Neural Network : These are mostly used in supervised learning. There are no feedback loops.

2) Feedback Neural Network: These are mainly for memory retention and are suitable for sequential data.

- **Applications of ANN:**

Speech Recognition, Signature Classification, Handwritten, Numbers recognition, Facial Recognition.

• Dataset

In this project we will be using the medical history of 1300~ people and predict the cost of health insurance premium based on these people's medical history.

This dataset consists of around 1300 odd people's medical history there are 7 columns which represent the person's history which we will be using in our predictions those are :

1. AGE
2. SEX
3. BMI(Body Mass Index)
4. Children (number of children they have)
5. Smoker(Whether they have a habit of smoking or not)
6. Region (Which region do they belong to) :

    Here we are considering 4 major regions

    - Southwest
    - Northwest
    - Southeast
    - Northeast

7. Expenses

The dataset has 1338 values in which the mean age of the people is 39, The proportion of people in dataset being men is 51% and women is 49%, The mean BMI of people being 30.7 which is overweight according NHS(national health service), Amongst the dataset where people smoke it came out that 20% are true and 80% are false and finally the distribution of people in the regions include southeast 27%, southwest 24% other being 49%, coming to expenses the mean expenses of the people in our dataset come around $13,300.

• Detail design of Features

A) Performing Exploratory Data Analysis

Exploratory Data Analysis refers to understanding the data before diving in and performing operations. Initially we try to discover underlying patterns, spot the anomalies present, gather as much as possible.

Libraries Used:

NumPy: For working with multi-dimensional arrays along with mathematical functions to operate on these arrays

Pandas: For data analysis and data manipulation. It offers operations and data structures for manipulating time series

Seaborn: It's the data visualization library we used. Used for the high-level drawing interface and statistical graphics.

Scikit-learn: It's the Data analysis library we used for predictive data analysis.

Tensorflow: It's an open source library developed by Google, used for Machine learning and Artificial Intelligence. It is known to provide significant focus on Deep learning which includes neural networks

Steps Performed :

We initially check for null values in our dataset.

Then we try to recognize patterns. In this step we found a relation between region and insurance charges. We can observe that the southeast region has higher BMI values when compared to northeast. From this we can assume that insurance charges may be higher in the southeast region.

We then do a grouping based on age and find the relation between age and expense.

Using seaborn library we plot the graphs to get an overview of how these factors impact the insurance cost.

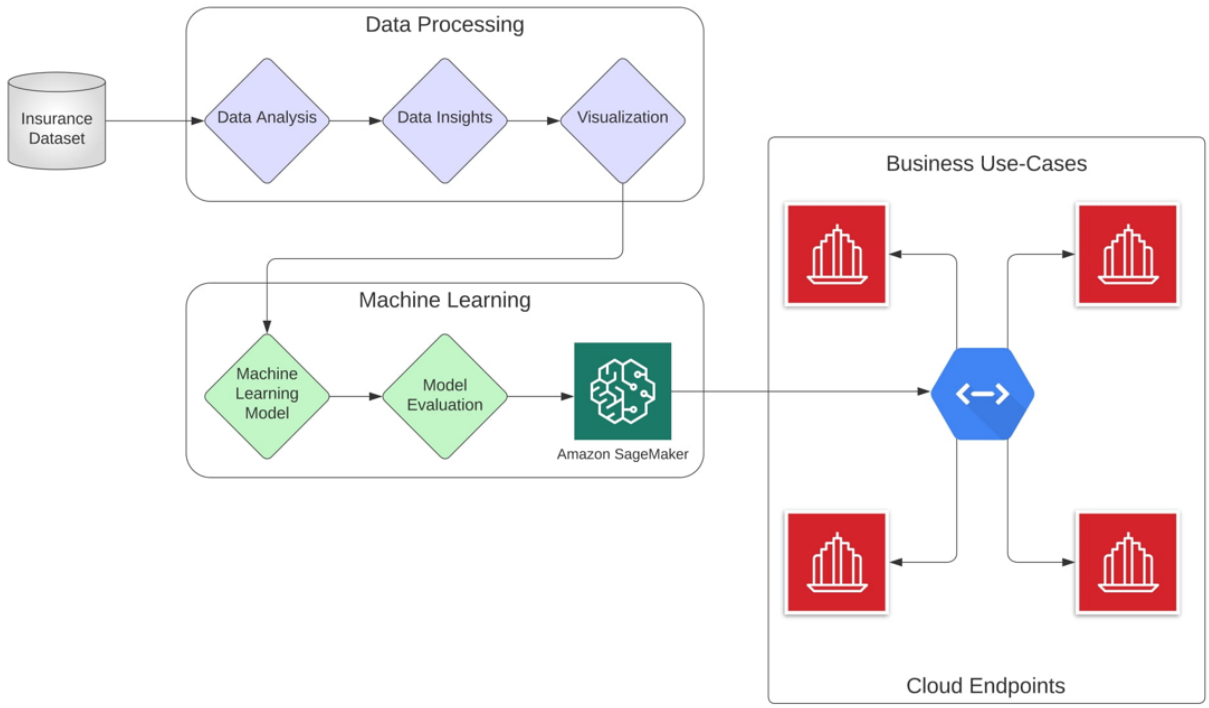
B) Predicting insurance costs

Now that we have observed underlying patterns and noted the observations in plot form, we try to predict the insurance costs. The input factors involved in predicting are: Age, Sex, BMI, children, smoker, region. We can observe that BMI and A person smokes or not , these two factors play a major role in cost prediction.

Analysis

- ➔ Dataset is to be pre-processed before performing any operations on it.
- ➔ Hence, we are performing exploratory data analysis on it with different techniques.
- ➔ Firstly, we are checking if there are any null values present in the data. Different insights we could see from the dataset itself which can give deeper understanding.
- ➔ The dataset visualization is performed using Seaborn which is one of the best available libraries to plot different data columns beautifully.

The architecture diagram is as follows.



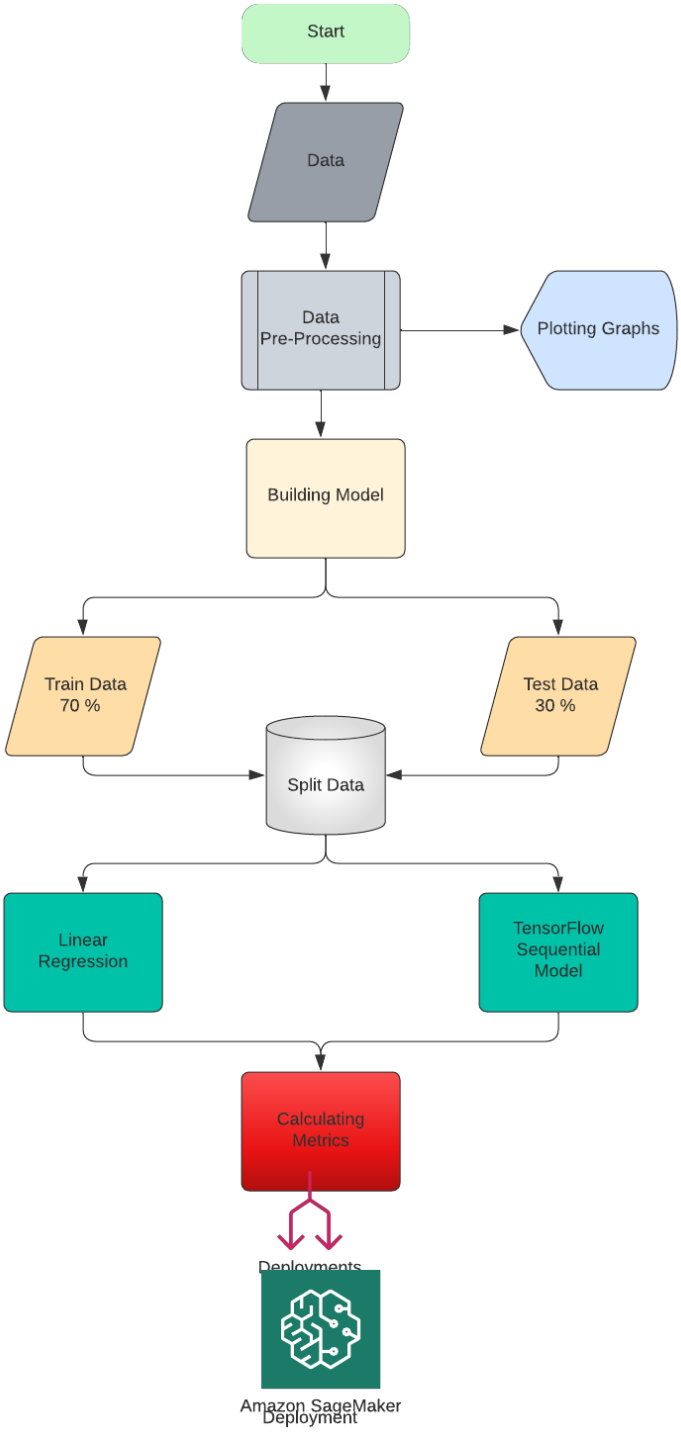
Machine Learning Models used:

- ★ Linear Regression
- ★ Tensorflow Sequential Model

Model Evaluation:

- ★ Mean absolute Error
- ★ Mean Square Error
- ★ Root Mean Square Error
- ★ R2

Workflow Diagram



• **Implementation**

- Importing libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

- Reading the dataset

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86
...	...	...	...	...	...	...	...
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

1338 rows × 7 columns

- Checking for null values

```
sns.heatmap(insurance.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c1d362710>
```

- Grouping regions to see relationship

```
region = insurance.groupby(by='region').mean()
region
# It reveals that the south east region has higher BMI values when compared to northeast
```

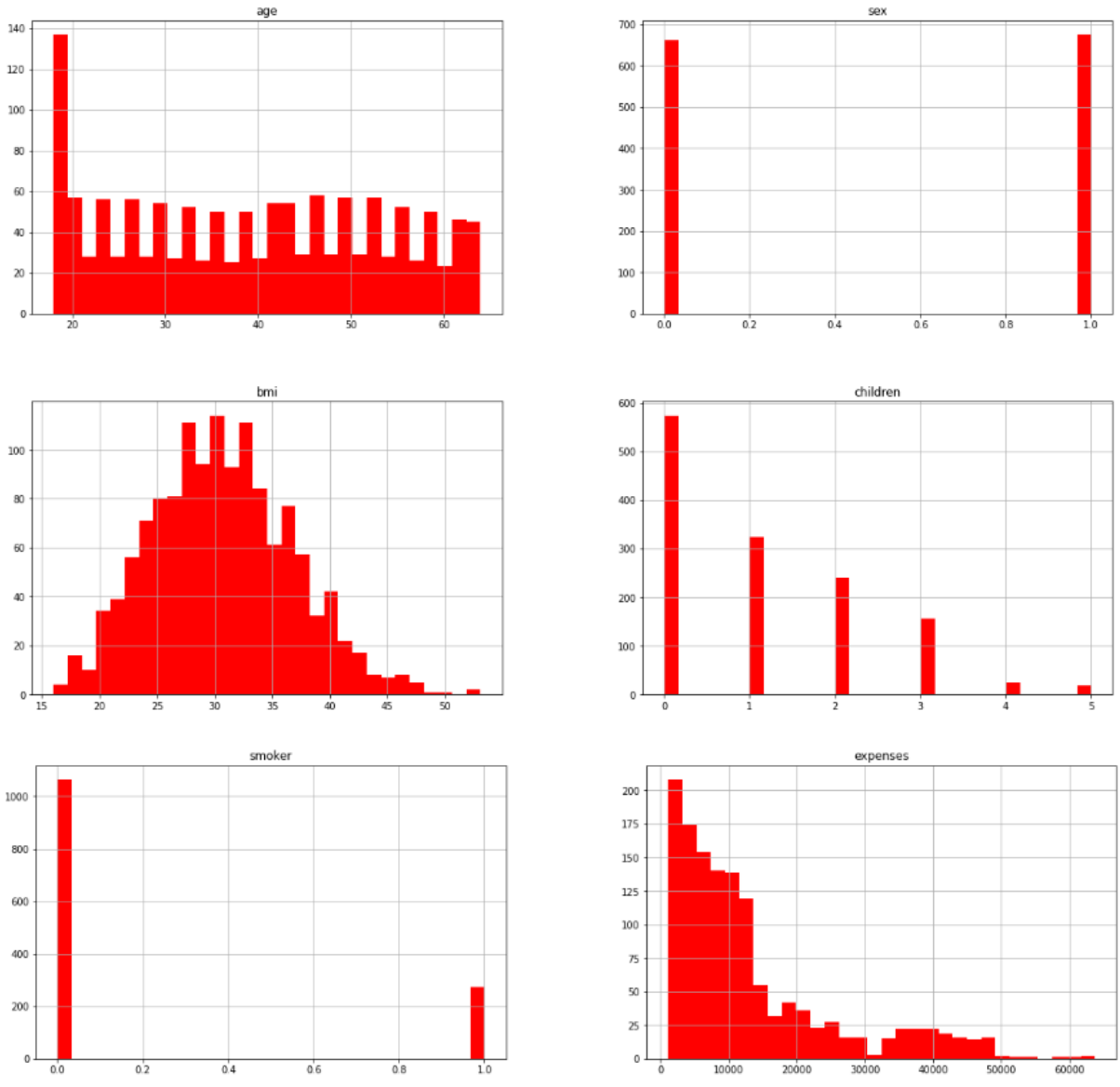
region	age	bmi	children	expenses
northeast	39.268519	29.176235	1.046296	13406.384691
northwest	39.196923	29.201846	1.147692	12417.575169
southeast	38.939560	33.359341	1.049451	14735.411538
southwest	39.455385	30.596615	1.141538	12346.937908

- Grouping by age

```
age = insurance.groupby(by='age').mean()
age
```

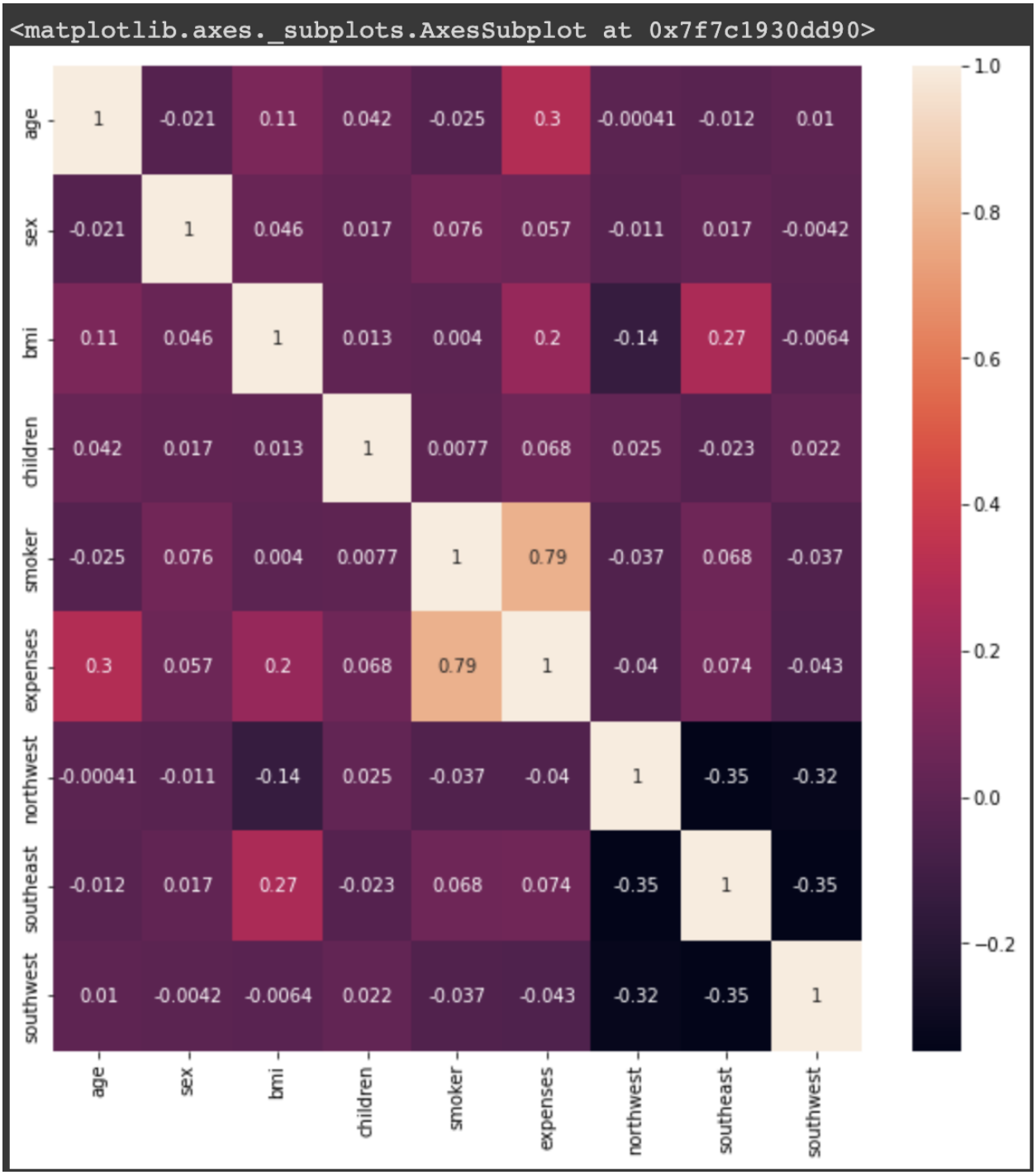
age	bmi	children	expenses
18	31.333333	0.449275	7086.217971
19	28.598529	0.426471	9747.909706
20	30.627586	0.862069	10159.697931
21	28.189286	0.785714	4730.464286
22	31.092857	0.714286	10012.932857
23	31.460714	1.000000	12419.820357
24	29.142857	0.464286	10648.015714
25	29.689286	1.285714	9838.365000
26	29.435714	1.071429	6133.825714
27	29.342857	0.964286	12184.701429
28	29.482143	1.285714	9069.187500
29	29.374074	1.259259	10430.159630
30	30.555556	1.555556	12719.111111

• **Preliminary Results**



Our preliminary results as you can see from the heat map show that there is a direct correlation between two factors: smoking and expenses. That is, the person who is a smoker has more medical expenses than the person who is a non-smoker, also between age and expenses. The age is directly proportional to expenses. More the age, the more the expenses and also between BMI and expenses, more the BMI greater the expenses.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c1930dd90>
```



- We import Standard Scalar from sklearn library StandardScalar standardizes features by removing the mean and rescaling it to a unit variance, then we import test_train_split from sklearn which we use to split our dataset into the training and testing dataset.
- We mention the test size parameter as 0.3 which means that we use our dataset in such a way that 30% is testing dataset and 70% is training dataset.

```
[93]: from sklearn.preprocessing import StandardScaler, MinMaxScaler

      scaler_x = StandardScaler()
      X = scaler_x.fit_transform(X)

      scaler_y = StandardScaler()
      y = scaler_y.fit_transform(y)
```

```
[94]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

```
[95]: X_train.shape
```

```
[95]: (936, 8)
```

```
[96]: X_test.shape
```

```
[96]: (402, 8)
```

- Now for performing regression we will import Linear Regression from sklearn's linear model and also import performance metrics like mean squared error and accuracy score,
- Create an instance of Linear regression and fit our training dataset into the model, we then calculate the accuracy of our model by testing it on the testing model, then we perform the prediction on the testing dataset.

```
[97]: from sklearn.linear_model import LinearRegression
      from sklearn.metrics import mean_squared_error, accuracy_score

      regresssion_model_sklearn = LinearRegression()
      regresssion_model_sklearn.fit(X_train, y_train)
```

```
[97]: ▾ LinearRegression
      LinearRegression()
```

```
[98]: regresssion_model_sklearn_accuracy = regresssion_model_sklearn.score(X_test, y_test)
      print("regression model accuracy is " + str(regresssion_model_sklearn_accuracy*100))
```

```
      regression model accuracy is 74.64827831648907
```

```
[99]: y_predict = regresssion_model_sklearn.predict(X_test)
```

```
[100]: y_predict_orig = scaler_y.inverse_transform(y_predict)
       y_test_orig = scaler_y.inverse_transform(y_test)
```

```
[101]: k = X_test.shape[1]
       n = len(X_test)
       n
```

```
[101]: 402
```

- Then we calculate the performance metrics for the Regression model which we have created we will be using 4 main performance metrics those are:
    1. RMSE(Root Mean Square Error)
    2. MSE(Mean Square Error)
    3. MAE(Mean Absolute Error)
    4. R2(R-squared) is a performance metric used to measure the variation in our regression model.

[102]:
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from math import sqrt

RMSE = float(format(np.sqrt(mean_squared_error(y_test_orig, y_predict_orig)),'.3f'))
MSE = mean_squared_error(y_test_orig, y_predict_orig)
```

[103]:
```
MAE = mean_absolute_error(y_test_orig, y_predict_orig)
r2 = r2_score(y_test_orig, y_predict_orig)
adj_r2 = 1-(1-r2) * (n-1)/(n-k-1)
```

[104]:
```
print('RMSE =',RMSE, '\nMSE =',MSE, '\nMAE =',MAE, '\nR2 =', r2, '\nAdjusted R2 =', adj_r2)
```

```
RMSE = 5871.895
MSE = 34479148.0
MAE = 4274.8813
R2 = 0.7464827930449673
Adjusted R2 = 0.7413221374326512
```

- Performing Artificial Neural Network on our Dataset, for that we first need to install the required libraries like tensorflow

105]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in ./.conda/envs/default/lib/python3.9/site-packages (2.11.0)
Requirement already satisfied: keras<2.12,>=2.11.0 in ./.conda/envs/default/lib/python3.9/site-packages (from tensorflow)
(2.11.0)
Requirement already satisfied: flatbuffers>=2.0 in ./.conda/envs/default/lib/python3.9/site-packages (from tensorflow) (2
2.11.23)
Requirement already satisfied: absl-py>=1.0.0 in ./.conda/envs/default/lib/python3.9/site-packages (from tensorflow) (1.
3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in ./.conda/envs/default/lib/python3.9/site-packages (from tensor
flow) (4.4.0)
Requirement already satisfied: wrapt>=1.11.0 in ./.conda/envs/default/lib/python3.9/site-packages (from tensorflow) (1.1
4.1)
Requirement already satisfied: setuptools in ./.conda/envs/default/lib/python3.9/site-packages (from tensorflow) (65.5.1)
Requirement already satisfied: numpy>=1.20 in ./.conda/envs/default/lib/python3.9/site-packages (from tensorflow) (1.23.
5)
```

- Now we install the other required libraries, these include keras,tensorflow, keras.layers and import Adam optimizer from keras.optimizers.

[106]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
```

- We now create a sequential model with 4 hidden layers and use an activation function of RELU(Rectified Linear Unit) for 3 layers and use the Linear activation function at the end and use Adam optimizer.

```
[107]: ANN_model = keras.Sequential()
ANN_model.add(Dense(50, input_dim = 8))
ANN_model.add(Activation('relu'))
ANN_model.add(Dense(150))
ANN_model.add(Activation('relu'))
ANN_model.add(Dropout(0.5))
ANN_model.add(Dense(150))
ANN_model.add(Activation('relu'))
ANN_model.add(Dropout(0.5))
ANN_model.add(Dense(50))
ANN_model.add(Activation('linear'))
ANN_model.add(Dense(1))
ANN_model.compile(loss = 'mse', optimizer = 'adam')
ANN_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 50)                450

 activation_8 (Activation)   (None, 50)                0

 dense_11 (Dense)            (None, 150)               7650

 activation_9 (Activation)   (None, 150)               0

 dropout_4 (Dropout)         (None, 150)               0

 dense_12 (Dense)            (None, 150)               22650

 activation_10 (Activation)  (None, 150)               0

 dropout_5 (Dropout)         (None, 150)               0

 dense_13 (Dense)            (None, 50)                7550

 activation_11 (Activation)  (None, 50)                0

 dense_14 (Dense)            (None, 1)                 51

=================================================================
Total params: 38,351
Trainable params: 38,351
Non-trainable params: 0
_________________________________________________________________
```

- Now we fit our model on the training dataset and set epochs to 100, batch size of 20 and split validation into 0.2

```
[108]: ANN_model.compile(optimizer='Adam', loss='mean_squared_error')

       epochs_hist = ANN_model.fit(X_train, y_train, epochs = 100, batch_size = 20, validation_split = 0.2)
```

```
Epoch 1/100
38/38 [==============================] - 3s 13ms/step - loss: 0.7304 - val_loss: 0.4560
Epoch 2/100
38/38 [==============================] - 0s 8ms/step - loss: 0.4466 - val_loss: 0.3254
Epoch 3/100
38/38 [==============================] - 0s 7ms/step - loss: 0.3499 - val_loss: 0.2671
Epoch 4/100
38/38 [==============================] - 0s 8ms/step - loss: 0.2580 - val_loss: 0.2449
Epoch 5/100
38/38 [==============================] - 0s 6ms/step - loss: 0.2391 - val_loss: 0.2801
Epoch 6/100
38/38 [==============================] - 0s 9ms/step - loss: 0.2409 - val_loss: 0.2369
Epoch 7/100
38/38 [==============================] - 0s 7ms/step - loss: 0.2400 - val_loss: 0.2505
Epoch 8/100
38/38 [==============================] - 0s 8ms/step - loss: 0.2139 - val_loss: 0.2317
```

- Now we evaluate the performance metric accuracy on the ANN model and we can see that the accuracy has improved from 74% in the Linear regression model to 84% in ANN.

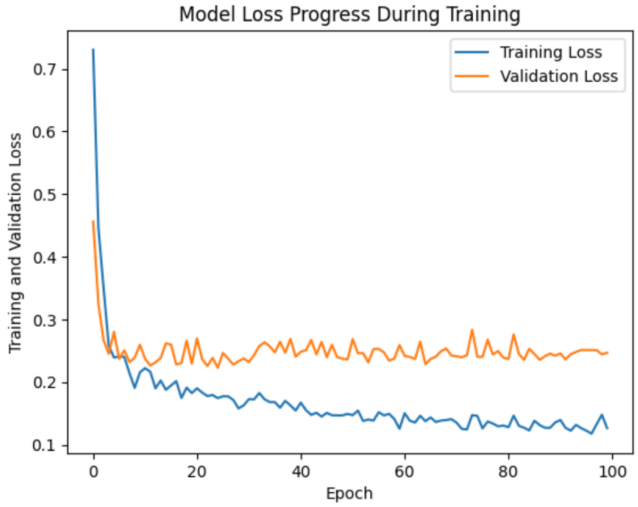
```
[109]: result = ANN_model.evaluate(X_test, y_test)
       accuracy_ANN = 1 - result
       print("Accuracy : {}".format(accuracy_ANN))
```

```
13/13 [==============================] - 0s 2ms/step - loss: 0.1594
Accuracy : 0.840594083070755
```

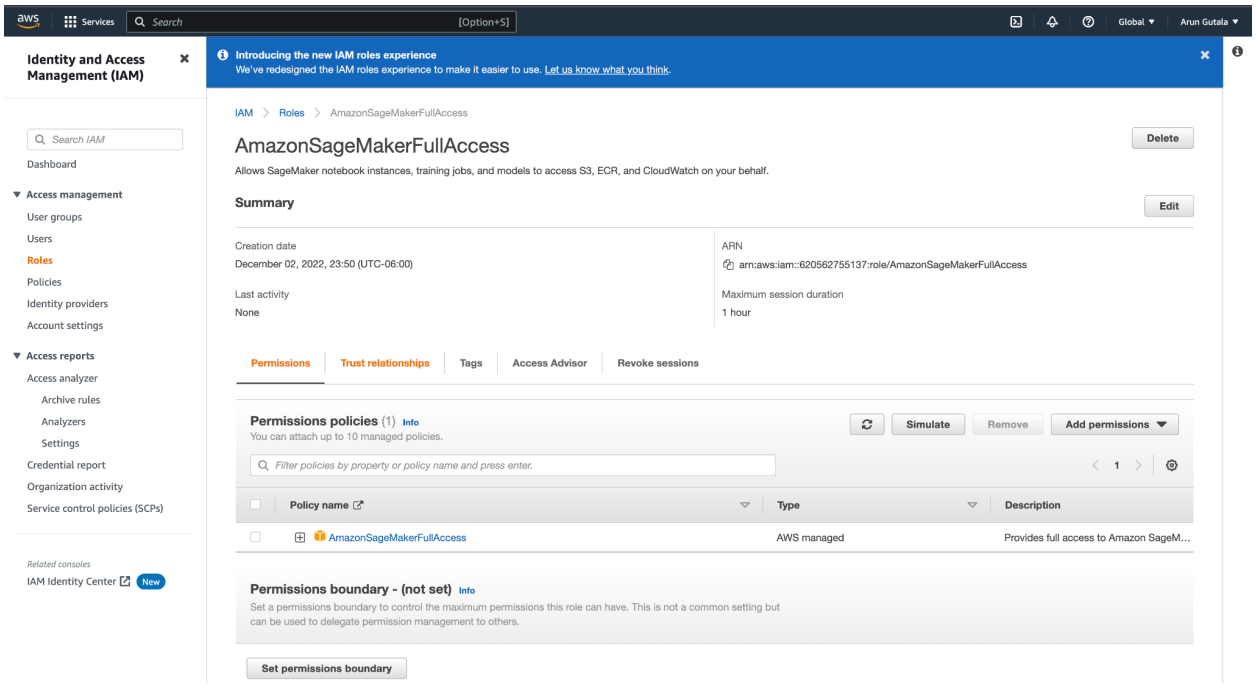
- Now, we plot a graph between Epochs and training and validation loss inorder to get an idea of how the model is performing during the training.

```
[110]: plt.plot(epochs_hist.history['loss'])
       plt.plot(epochs_hist.history['val_loss'])
       plt.title('Model Loss Progress During Training')
       plt.xlabel('Epoch')
       plt.ylabel('Training and Validation Loss')
       plt.legend(['Training Loss', 'Validation Loss'])
```

```
[110]: <matplotlib.legend.Legend at 0x7f1f54220f10>
```



- The project is now deployed onto Sagemaker Studio Lab where we are performing the final operations.
- To enable the sagemaker endpoints, the aws configuration is to be configured with access id and key to enable Sagemaker lab to access AWS resources.
- These keys are saved inside a config folder which is a subfolder in aws root directory.
- It also requires an AWS IAM role which enables sagemaker to access different resources by adding permissions to it.
- Here, we are calling it as 'AmazonSageMakerFullAccess'



- **Final Results**

Here we are comparing two different machine learning models which are giving the prediction results. Regression deals mostly with LinearDependencies and Neural Networks deal with non-linearities. Neural Networks is a deep learning method which relies on big datasets. Regression models need less data and they perform well only with smaller datasets whereas Neural Networks can work on immensely big datasets.Neural Networks fit better for historical data. We should rely on the prediction power to avoid any chances of overfitting. Neural Networks have sophisticated architecture with included activation functions such as sigmoid, ReLu which helps in predicting accurate models.

If looking for optimisation, regression models can be easy over Neural Networks models.

Project Management

Implementation status report

Work Completed

In this Increment we have done the following:

- ➢ Performing Exploratory Data Analysis, Predicting insurance costs.
- ➢ Using the heat map we have found a direct correlation between the input factors and expenses.
- ➢ Model Building , calculating metrics for our model.
- ➢ Prediction and Model accuracy is increased by supplementing with advanced technologies like ANN.
- ➢ Hosted the project on AWS Cloud services and exposed endpoints of the model.

Description

• Responsibility/ Contributions (members/Task/percentage)

Member Name	Contribution description	Overall Contribution
Hari Vamsi Krishna S	Objectives, Features, Dataset and Code(Dataset Implementation)	40%
Arun Sai Kumar Gutala	Dataset description & implementing code for seaborn graphs, Analysis, Implementation	30%
Sindhuja Chepuri	Related work, Detail Design of Features, Project management, Preliminary Results	30%

• Issues/Concerns

The training data which we provided may not be 100% authentic and there could be false information when recording the demographics.

5. References

- Ahmed, A. M., Rizaner, A. & Ulusoy, A. H., 2018. A Decision Tree Algorithm Combined with Linear Regression for Data Classification.
- Health Insurance Claim Prediction Using Artificial Neural Networks [Sam Goundar (The University of the South Pacific, Suva, Fiji), Suneet Prakash (The University of the South Pacific, Suva, Fiji), Pranil Sadal (The University of the South Pacific, Suva, Fiji) and Akashdeep Bhardwaj (University of Petroleum and Energy Studies, India)]
- Burns, E. et al., 2019. Considerations for Predictive Modeling in Insurance Applications, s.l.: Society of Actuaries.
- Open-source Insurance dataset available on Kaggle. [https://www.kaggle.com/datasets/awaiskaggler/insurance-csv]