The University of Memphis
Computer Science
COMP 7/8150-Fundamentals of Data Science
Project Proposal: **Health Insurance Premium at PSI**
Vinay Badam, Nikhil Banka, Data Science, Fall 2022

PROJECT GOAL AND SCOPE

The goal of this project is to predict the health insurance premiums of customers of PSI six months in advance. The solution will be one python program that predicts the premium with an average relative error of at most 20%.

Problem:
Instance: In the pandemic Covid-19, every individual chooses fewer Deductibles by paying high premiums.
Solution: By taking precautions, maintaining social distancing, and eating healthy food can be prevented from Covid-19 up to some extent. Decrease in premium price.

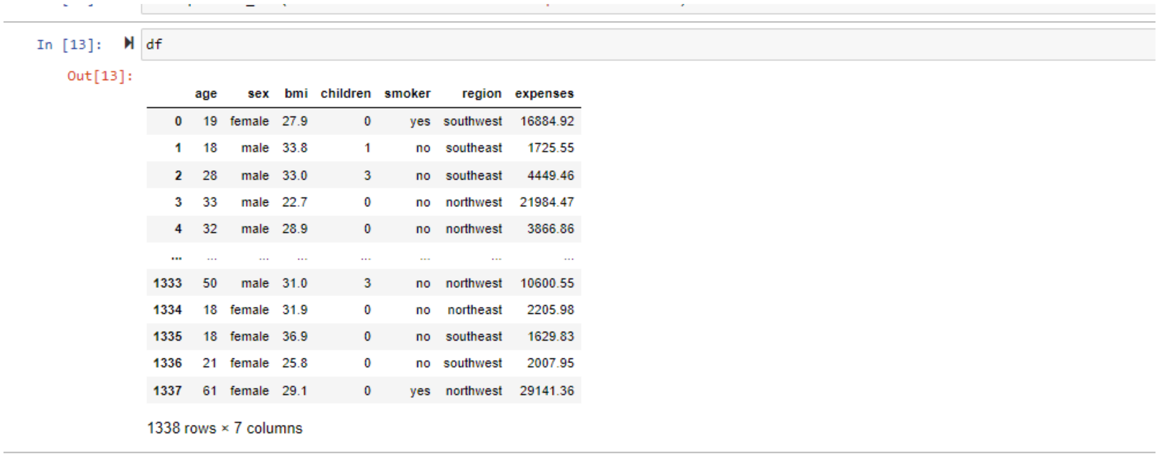
BACKGROUND/LIT REVIEW

{VB} In the event of a major illness or accident, it safeguards your financial situation. Health insurance is one form of policy that helps people budget for and pay for medical costs. Insurance companies use a person's age, physical condition, family situation, and past medical bills to predict their future medical costs to decide how much to charge for coverage.

The full population under consideration for the project would be clients of the insurance provider PSI. This dataset [2] contains data on the insurer, their dependents, their medical costs over a year, etc. The prediction is based on the dataset's other parameters such as BMI, smoker, age, etc. In the dataset, we can also see the previous expenses paid by clients for the previous term. The dataset [2] contains the following data items:

1. Age
2. Sex
3. BMI: Body Mass Index of the insurer
4. Children: determines if the insurer has any children to be covered.
5. Smoker: Determines whether he/she is a smoker or non-smoker
6. Region: Residence of the insurer
7. Charges: Yearly medical charge

The dataset contains 1338 records and has a dimensionality of 7



```
In [13]:  df
Out[13]:
```

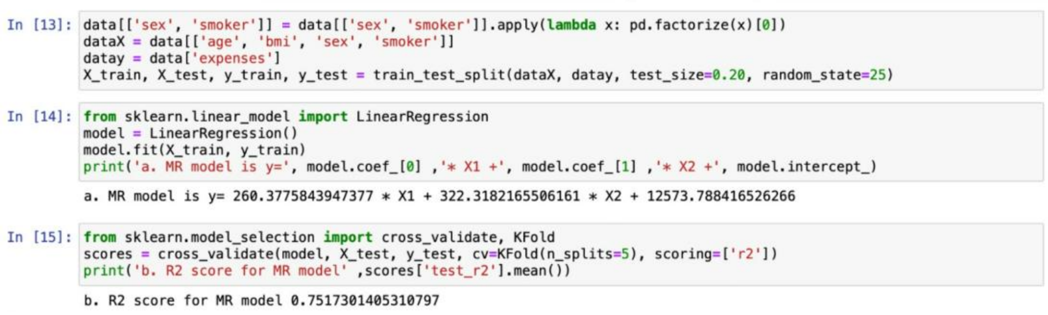
	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86
...	...	...	...	...	...	...	...
1333	50	male	31.0	3	no	northwest	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

Findings:

1) From the above Data, we can find Expenses is directly proportional to the Age factor.
2) From the data, 1338 Rows & 7 columns are considered to train the dataset in the proposed model.

{NB} The articles [3] and [4] worked on the insurance prediction problem using linear regression based on the datasets [2] and data available from the dataset [2] having required data of insurance firm clients which resulted in a relative error or mean squared error for the data to be higher than expected. The author [4] divided the features into numerical and nominal features. These nominal features were converted into factors with a numerical value. To understand the data, the author plotted the features' dispersion graphs and then calculated the mean and median for age, BMI, and expenses. The author found that there is a very slight variation in the mean and median for age and BMI but for expenses, there is a high variation. The author then has done a scatter plot between age and expenses concerning BMI to understand why people with low age have high BMI and high age with low BMI. So, the three features, age, BMI, and smoker have a relationship with expenses and give satisfactory MR value.



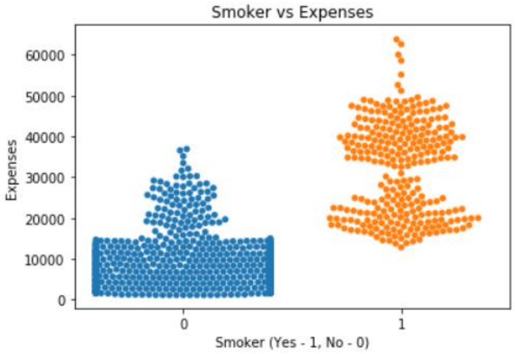
```
In [13]: data[['sex', 'smoker']] = data[['sex', 'smoker']].apply(lambda x: pd.factorize(x)[0])
         dataX = data[['age', 'bmi', 'sex', 'smoker']]
         datay = data['expenses']
         X_train, X_test, y_train, y_test = train_test_split(dataX, datay, test_size=0.20, random_state=25)

In [14]: from sklearn.linear_model import LinearRegression
         model = LinearRegression()
         model.fit(X_train, y_train)
         print('a. MR model is y=', model.coef_[0] ,'* X1 +', model.coef_[1] ,'* X2 +', model.intercept_)

         a. MR model is y= 260.3775843947377 * X1 + 322.3182165506161 * X2 + 12573.788416526266

In [15]: from sklearn.model_selection import cross_validate, KFold
         scores = cross_validate(model, X_test, y_test, cv=KFold(n_splits=5), scoring=['r2'])
         print('b. R2 score for MR model' ,scores['test_r2'].mean())

         b. R2 score for MR model 0.7517301405310797
```

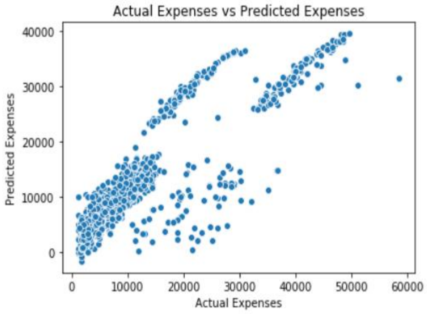
Findings:

1) We can see from the above code; we considered Sex & whether Smokers or not are dependent variables which is the main factor to decide the premium.
2) Applied Linear Regression & Multiple Regression, Achieved an R2 score is 0.75 for MR which is best when compared to Linear Regression

CASE STUDY

Venkat Murali [4] presented a model for insurance premium prediction [4] using linear regression. Here Linear regression is used to predict the value of a variable based on the value of another variable. The variable we are trying to predict is called the dependent variable which is Expenses and the variable used to predict the other variable's value is called the independent variable which is sex, age, and BMI. Here the author used basic linear regression to predict the insurance premium. The author of this article used a small dataset from the entire dataset and then created a linear regression training model and fed 30% of the data and then plotted a graph to see the actual versus predicted expenses and r squared value of 0.732.

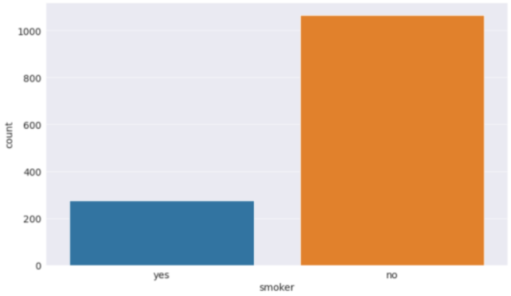




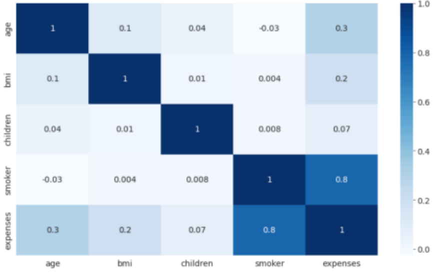
From the above diagram, we can observe that Smokers (yes or no) are on X -the axis and Expenses on Y-axis, when we plot the graph, we can conclude that expenses are very high for smokers when compared to non-smokers.

From the above paper, I've considered the entire dataset (1338 records) after analyzing the data and finding the relationship between different features. I first plotted each variable's significance from the dataset before creating a model.

Number of insurers by smoking



Correlation of heat map



Problem Definition:

A health insurance premium is an upfront payment made on behalf of an individual or family to keep their health insurance policy active. By using a linear regression method.

My findings are smokers are affected with more health problems compared to non-smokers with fewer problems and in the correlation matrix, we can summarize the relationship between dependent and independent variables.

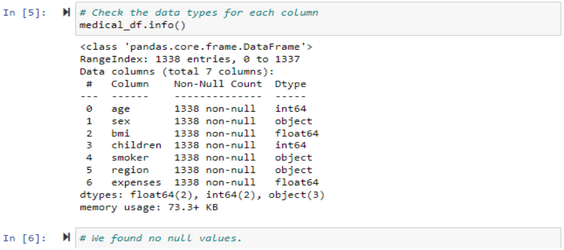
This case study helps to understand how health insurance premium varies based on different independent variables i.e BMI, Age, Sex, and Expenses. Considering this case study, we completed our project.

TAKE-HOME DELIVERABLE

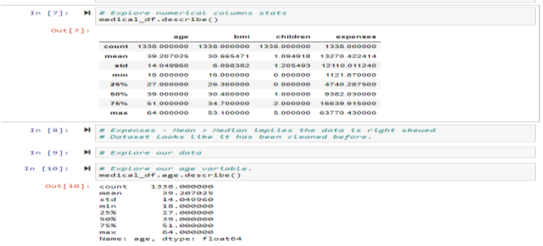
The deliverables for this project will be one python program that predicts the premium with an average relative error of at most 20% over the whole population as estimated from the dataset.

The solution will be one python program that predicts the premium with an average relative error of at most 20% using Linear Regression methodology. Medical expenses are the dependent variable, and all other variables age, BMI, smoking, children, and gender are independent variables. The prediction of the premium charges is dependent on these variables, so to formulate the relationship between these independent variables and expenses, From the summary of the linear model, we can notice that regression coefficients age, BMI, smoker, and children are significant. From the summary of the linear model, we can observe that the multiple R squared value is 0.800 which tells us that the model accounts for 80% of the variance in the premium.

I have performed the following steps to solve the problem
1) Collection of Dataset: I have collected data from the Kaggle with 1338 observations and 7 columns.
2) Data preprocessing: Data preprocessing is the strategy used to convert the raw data into a comprehensible collection of data. As the data obtained from the dataset have different forms of data it should be converted into a format that can be understood by the machine to perform further actions.
3) Applying the Algorithm: Applying the algorithm by considering the Train Data and Test Data.
4) Train the Algorithm: Considered 80% of Data from preprocessing into the algorithm with 80% of train data and 20%of test data.
5) Test the model: Test the model with all inputs from the algorithm Linear regression, random forest, and decision tree.
6) Deploy: After getting satisfactory results we are going to deploy the project into real-world scenarios to get the desired outputs.

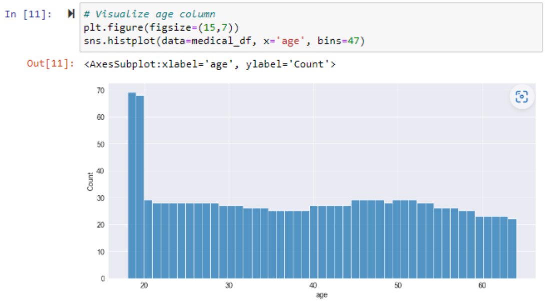


```
In [5]:  # Check the data types for each column
         medical_df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 1338 entries, 0 to 1337
         Data columns (total 7 columns):
          #   Column    Non-Null Count  Dtype
         ---  ------    --------------  -----
          0   age       1338 non-null   int64
          1   sex       1338 non-null   object
          2   bmi       1338 non-null   float64
          3   children  1338 non-null   int64
          4   smoker    1338 non-null   object
          5   region    1338 non-null   object
          6   expenses  1338 non-null   float64
         dtypes: float64(2), int64(2), object(3)
         memory usage: 73.3+ KB

In [6]:  # We found no null values.
```

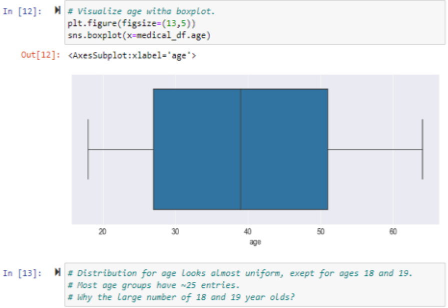


```
In [7]:  # Explore numerical columns stats
         medical_df.describe()

Out[7]:
                 age          bmi     children      expenses
count    1338.000000  1338.000000  1338.000000   1338.000000
mean       39.207025    30.665471     1.094918  13270.422414
std        14.049960     6.098382     1.205493  12110.011240
min        18.000000    16.000000     0.000000   1121.870000
25%        27.000000    26.300000     0.000000   4740.287500
50%        39.000000    30.400000     1.000000   9382.030000
75%        51.000000    34.700000     2.000000  16639.915000
max        64.000000    53.100000     5.000000  63770.430000

In [8]:  # Expenses - Mean > Median implies the data is right skewed
         # Dataset looks like it has been cleaned before.

In [9]:  # Explore our data

In [10]: # Explore our age variable.
         medical_df.age.describe()

Out[10]: count    1338.000000
         mean       39.207025
         std        14.049960
         min        18.000000
         25%        27.000000
         50%        39.000000
         75%        51.000000
         max        64.000000
         Name: age, dtype: float64
```

- From the above we can see there are no Null values in the data and calculated mean, median, and standard deviation values.



```
In [11]:  # Visualize age column
          plt.figure(figsize=(15,7))
          sns.histplot(data=medical_df, x='age', bins=47)

Out[11]:  <AxesSubplot:xlabel='age', ylabel='Count'>
```

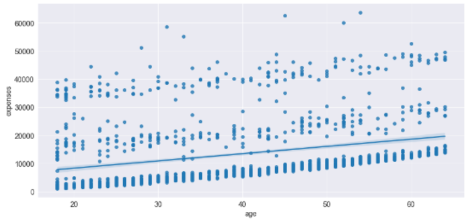


```
In [12]:  # Visualize age witha boxplot.
          plt.figure(figsize=(13,5))
          sns.boxplot(x=medical_df.age)

Out[12]:  <AxesSubplot:xlabel='age'>
```

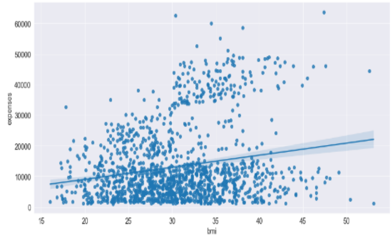
```
In [13]:  # Distribution for age looks almost uniform, exept for ages 18 and 19.
          # Most age groups have ~25 entries.
          # Why the large number of 18 and 19 year olds?
```

from the above graphs, we can conclude that there are a large number of people under the age of 20 when compared to ages greater than 20

```
In [27]:  # Now we take a look at the relationship between variables.

In [28]:  # Explore relationship between AGE and CHARGES
          plt.figure(figsize=(15,7))
          sns.regplot(x=medical_df.age, y=medical_df.expenses)

Out[28]:  <AxesSubplot:xlabel='age', ylabel='expenses'>
```

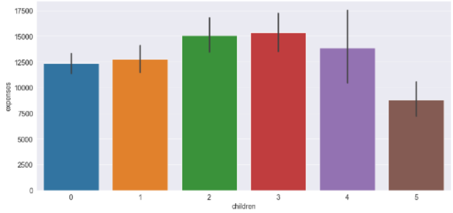


```
In [29]:  # Explore relationship between BMI and CHARGES
          plt.figure(figsize=(15,7))
          sns.regplot(x=medical_df.bmi, y=medical_df.expenses)

Out[29]:  <AxesSubplot:xlabel='bmi', ylabel='expenses'>
```

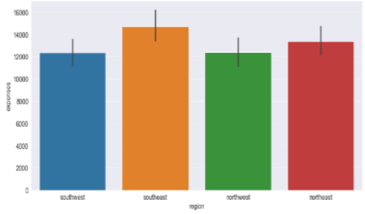


Findings: Scatter plot clearly states that when age is increasing expenses also increase but has three different groups of expenses irrespective of BMI. Hence, BMI is not influencing expenses with Age.

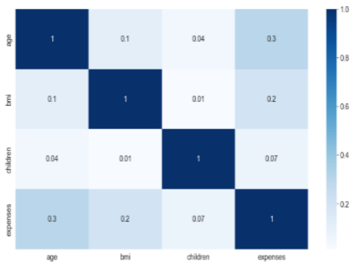
```
In [34]:  # Explore other expenses relationships.
          plt.figure(figsize=(15,7))
          sns.barplot(x=medical_df.children, y=medical_df.expenses)

Out[34]:  <AxesSubplot:xlabel='children', ylabel='expenses'>
```



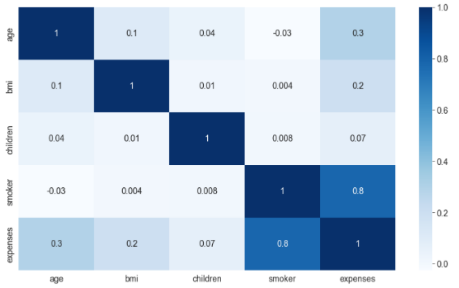
```
In [35]:  plt.figure(figsize=(15,7))
          sns.barplot(x=medical_df.region, y=medical_df.expenses)

Out[35]:  <AxesSubplot:xlabel='region', ylabel='expenses'>
```



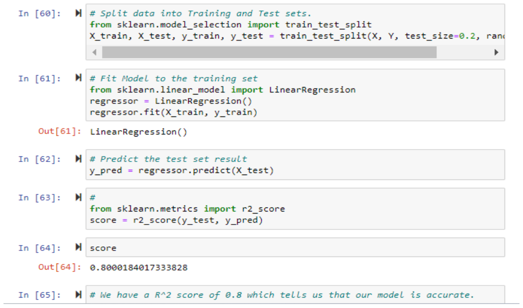
```
In [37]:  # Find the correlation coefficient of our columns.

In [38]:  medical_df.expenses.corr(medical_df.age)

Out[38]:  0.29900819228508263

In [39]:  medical_df.expenses.corr(medical_df.bmi)

Out[39]:  0.19857625501893197

In [40]:  medical_df.expenses.corr(medical_df.children)

Out[40]:  0.06799823000532802

In [41]:  # visualize correlation using a heatmap
          plt.figure(figsize=(14,8))
          cor = medical_df.corr()
          sns.heatmap(cor, annot=True, fmt='.1g', cmap='Blues')

Out[41]:  <AxesSubplot:>
```



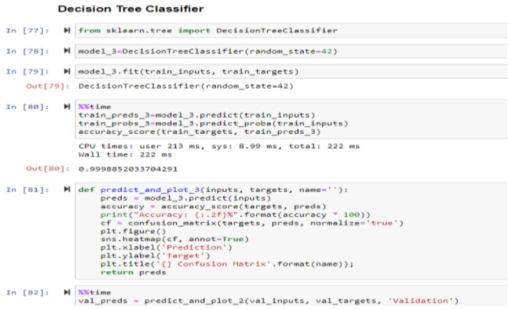
```
In [44]:  medical_df.expenses.corr(smoker_numeric)

Out[44]:  0.7872514298985548

In [45]:  # create a df with smoker value as numeric
          smoker_num_df = medical_df.replace(['yes', 'no'], [1, 0])

In [46]:  smoker_num_df.expenses.corr(smoker_num_df.smoker)

Out[46]:  0.7872514298985548

In [47]:  plt.figure(figsize=(14,8))
          cor = smoker_num_df.corr()
          sns.heatmap(cor, annot=True, fmt='.1g', cmap='Blues')

Out[47]:  <AxesSubplot:>
```



As you can see above, we obtain the heatmap of correlation among the variables. The color palette in the side represents the amount of correlation among the variables. The lighter shade represents a high correlation.

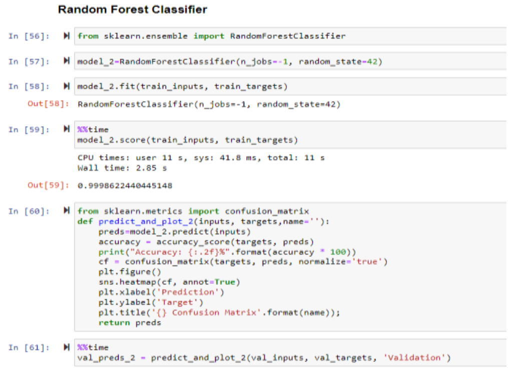


```
In [60]:  # Split data into Training and Test sets.
          from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rand

In [61]:  # Fit Model to the training set
          from sklearn.linear_model import LinearRegression
          regressor = LinearRegression()
          regressor.fit(X_train, y_train)

Out[61]:  LinearRegression()

In [62]:  # Predict the test set result
          y_pred = regressor.predict(X_test)

In [63]:  #
          from sklearn.metrics import r2_score
          score = r2_score(y_test, y_pred)

In [64]:  score

Out[64]:  0.8000184017333828

In [65]:  # We have a R^2 score of 0.8 which tells us that our model is accurate.
```



Decision Tree Classifier

```
In [77]:  from sklearn.tree import DecisionTreeClassifier

In [78]:  model_3=DecisionTreeClassifier(random_state=42)

In [79]:  model_3.fit(train_inputs, train_targets)

Out[79]:  DecisionTreeClassifier(random_state=42)

In [80]:  %%time
          train_preds_3=model_3.predict(train_inputs)
          train_probs_3=model_3.predict_proba(train_inputs)
          accuracy_score(train_targets, train_preds_3)

          CPU times: user 213 ms, sys: 8.99 ms, total: 222 ms
          Wall time: 222 ms

Out[80]:  0.9998852033704291

In [81]:  def predict_and_plot_3(inputs, targets, name=''):
              preds = model_3.predict(inputs)
              accuracy = accuracy_score(targets, preds)
              print("Accuracy: {:.2f}%".format(accuracy * 100))
              cf = confusion_matrix(targets, preds, normalize='true')
              plt.figure()
              sns.heatmap(cf, annot=True)
              plt.xlabel('Prediction')
              plt.ylabel('Target')
              plt.title('{} Confusion Matrix'.format(name));
              return preds

In [82]:  %%time
          val_preds = predict_and_plot_2(val_inputs, val_targets, 'Validation')
```



Random Forest Classifier

```
In [56]:  from sklearn.ensemble import RandomForestClassifier

In [57]:  model_2=RandomForestClassifier(n_jobs=-1, random_state=42)

In [58]:  model_2.fit(train_inputs, train_targets)

Out[58]:  RandomForestClassifier(n_jobs=-1, random_state=42)

In [59]:  %%time
          model_2.score(train_inputs, train_targets)

          CPU times: user 11 s, sys: 41.8 ms, total: 11 s
          Wall time: 2.85 s

Out[59]:  0.9998622440445148

In [60]:  from sklearn.metrics import confusion_matrix
          def predict_and_plot_2(inputs, targets,name=''):
              preds=model_2.predict(inputs)
              accuracy = accuracy_score(targets, preds)
              print("Accuracy: {:.2f}%".format(accuracy * 100))
              cf = confusion_matrix(targets, preds, normalize='true')
              plt.figure()
              sns.heatmap(cf, annot=True)
              plt.xlabel('Prediction')
              plt.ylabel('Target')
              plt.title('{} Confusion Matrix'.format(name));
              return preds

In [61]:  %%time
          val_preds_2 = predict_and_plot_2(val_inputs, val_targets, 'Validation')
```

After applying and deploying the algorithm we can conclude that model is accurate at 80%, we can say that This would indicate that half of the dependent variable variance is explained by the model's independent variables.

Next, regression trees divide a data set into smaller subgroups and then fit a simple constant to each of the subgroup's observations. Based on the different predictors, successive binary partitions (also known as recursive partitioning) are used to partition the data. The average response values for all observations in that subgroup are used to calculate the constant to forecast.
The value of MSE is 152875 which is higher than the linear regression model. Therefore, we are implementing another model to have better accuracy and lesser mean squared value. I choose random forest because it is more robust, and it can capture those non-linear features in my data. By default, the test set MSE is 30381 which improved from both linear regression and decision tree. I've ordered the importance of the variables, “smoker" is the most important variable across all the trees considered in the random forest, followed by "age", "BMI" and "children

Comparing the models obtained from Linear Regression, Decision tree, and Random Forest, the better fit is seen for Random Forest and then the linear model. For assessment, we’ve found that the multiple r squared value is 0.80 which indicates that the random forest model has at most average relative error of 20%.

Real-world scenario:
For example, you may pay a premium of Rs. 5000 each year for a health insurance cover of Rs. 200,000/- so that if God forbid, you fall ill and need to be hospitalized in that year, the insurance provider company will bear the cost of hospitalization, etc. for up to Rs. 200,000. Now if you are wondering how the company can bear such high hospitalization costs when it charges a premium of only Rs. 5000/-, that is where the concept of probabilities comes into the picture. For example, like you, there may be 100 customers who would be paying a premium of Rs. 5000 every year, but only a few of them (say 2-3) would get hospitalized that year and not everyone. This way everyone shares the risk of everyone else.

Future Possible Work:

- Betterment of results using different hyperparameters for tuning
- Implementing more models to gain better results
- Using the same method to predict responses for various other kinds of insurance.
- Combining all the processing of insurance offer advertisements and achieving the best customers for the same

For assessment, we've found that the multiple r squared value is 0.80 which indicates that the random forest model has at most average relative error of 20%.

REFERENCES

[1] Centers for Medicare Services (CMS)
https://data.cms.gov/search?keywords=HEALTH%20INSURANCE&sort=Relevancy

[2] https://www.kaggle.com/code/mariapushkareva/medical-insurance-cost-with-linear-regression

[3] https://www.kaggle.com/datasets/hhs/health-insurance-marketplace

[4] Insurance Premium Prediction - Linear Regression | Kaggle