Assignment 1 CuiJiaying 3036046191

February 28, 2023

1 Look at the big picture

1.1 Clearly Clarify Tasks

- In this case analysis, my mission is to use **Linear Regression** to build a prediction engine.
- And as a data analyst, I should also observe the insurance data performance and to make related recommendations.

1.2 The Framework to handle this task

- Use the supervised learning
  - Clarify the **"past claims"** as the y, the result variables
  - Use the previous seven variables as the variables
  - Check the Correlations of different variables before use the model to make whole predictions process more reliable
- Use the Linear Regression
  - Before using the model, the first step is to handle the data
  - Data Processing includes
    * Check and process repeated values
    * Check and process null values
    * Check and process strange values
    * Change the Categorical Attributes
    * Standardize the data
  - Check the Correlations
  - Split the Train Data and Test Data
  - Use the Train Data to train the model
- Choose the Performance Measures
  - Use the r2_score, mean_squared_error, mean_absolute_error to evaluate the model

2 Get the Data

- In this case, I use the pandas to import dataset to read and analyse
- use the **"pd.read_csv()"** function to import data
- When processing the data for the first time, I found that the last column **"past claims"** contains thousands of separating commas, if only read the file, the value of this column will become object difficult to handle. Therefore, I add a condition parameters, **"thousands=',' "** to automatically transfer this column into float

[1]:
```python
import pandas as pd
insurance=pd.read_csv('InsuranceDataset.csv',thousands=',')
# use thousands=',',if not the past claims will be object
```

2.1 take a quick look on the data structure

2.1.1 Use the info() method to check

- Use the **info()** method to get a summary description of the data structure
- Get the overall idea about the dataset

[2]:
```python
insurance.info() # use the info() method check the data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1341 entries, 0 to 1340
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          1338 non-null   float64
 1   sex          1338 non-null   object
 2   home         1338 non-null   object
 3   bmi          1338 non-null   float64
 4   children     1335 non-null   float64
 5   smoker       1338 non-null   object
 6   drinking     1338 non-null   object
 7   past claims  1338 non-null   float64
dtypes: float64(4), object(4)
memory usage: 83.9+ KB
```

- **.info() Findings**
  - Data shows that in this DataFrame, there are **8 Columns**

  - **"age,bmi,children,past claims"** are float
  - **"sex,home,smoker,drinking"are object
  - Pay attention to the four non-numerical data, remember to process these data later
  - **children** only has 1335 non-null values, it seems children has to handle the **null value**, I will process it later in the Data Processing

2.1.2 Use the describe() method to check

- Use **the describe()** method to show the summary statistics of attributes
- I add the **include='all'** to check both the numeric and non-numeric situations

[3]:
```python
insurance.describe(include='all')
```

[3]:
```
                age   sex      home          bmi     children smoker  \
count   1338.000000  1338      1338  1338.000000  1335.000000   1338
unique          NaN     2         4          NaN          NaN      2
top             NaN  male  South NT          NaN          NaN     no
```

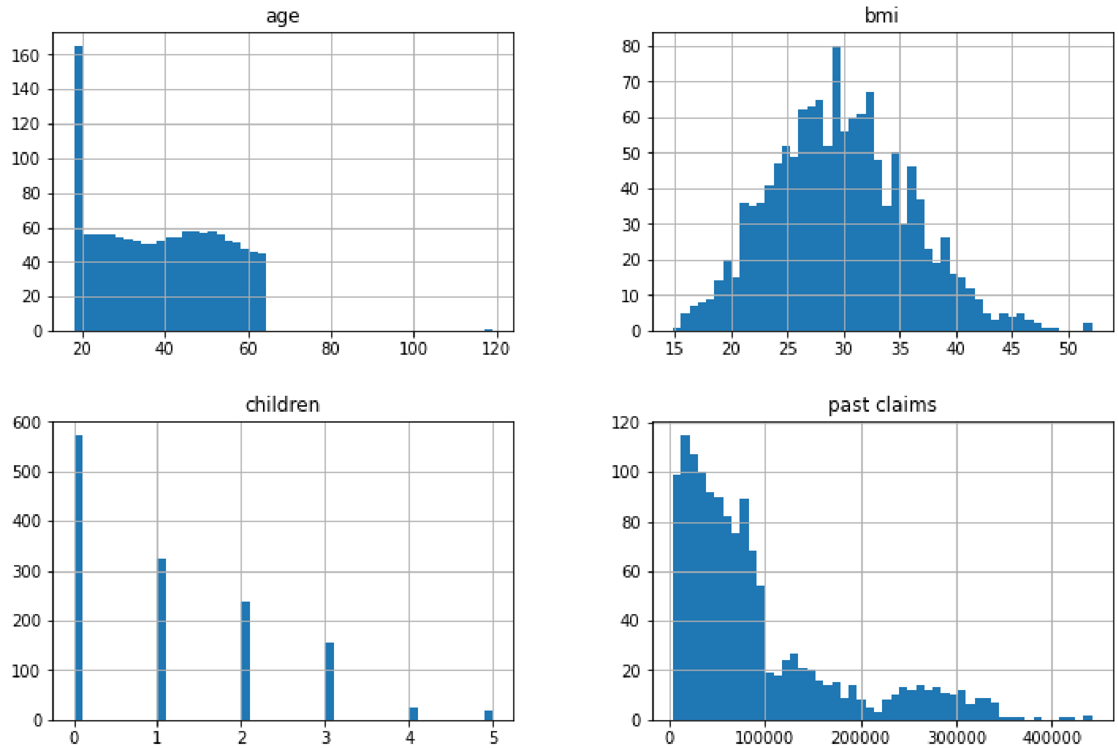
```
freq            NaN    676       364          NaN          NaN   1064
mean      39.281764    NaN       NaN    29.738341     1.093633    NaN
std       14.207480    NaN       NaN     6.109329     1.205092    NaN
min       18.000000    NaN       NaN    14.800000     0.000000    NaN
25%       27.000000    NaN       NaN    25.400000     0.000000    NaN
50%       39.000000    NaN       NaN    29.500000     1.000000    NaN
75%       51.000000    NaN       NaN    33.600000     2.000000    NaN
max      119.000000    NaN       NaN    52.100000     5.000000    NaN

          drinking    past claims
count         1338    1338.000000
unique           3            NaN
top     occasional            NaN
freq           800            NaN
mean           NaN   90388.195815
std            NaN   84782.257933
min            NaN    3374.000000
25%            NaN   30353.750000
50%            NaN   63390.000000
75%            NaN  113611.000000
max            NaN  442160.000000
```

- **method()** Findings
  - **age:** data shows that average age is 39, min 18, max 119, it seems the max age doesn't meet the real case, it needs to delete the strange value
  - **sex:** male is larger than female
  - **home:** South NT is the largest areas in this dataset, and we have four areas in total
  - **bmi:** average BMI is 30, according to the international standards, **BMI over 25 is fat**, data shows that in this case over 75% is fat
  - **children:** data shows that each person on average has one child, max has 5 kids
  - **smoker:** in this case, non-smoker is larger than smoker
  - **drinking:** there are three degrees of drinking, occasional is the largest
  - **past claims:** it shows that "past claims" std is too large, maybe it needs to check the distribution and process the data.
  - **Categorize the above variables**
    * **"age","sex","bmi","children" is a person's internal features;**
    * **"home"is external features;**
    * **"smoker","drinking" is behavioral features**

2.2 Visualize the Data Distribution

2.2.1 Use the hist chart to check the distribution of the numeric data

[4]:
```python
import matplotlib.pyplot as plt
insurance.hist(bins=50, figsize=(12, 8))
plt.show()
```



- **Hist Chart Findings**
  - **Age:** It shows that "around 20" is the biggest part, except that it almost evenly distribution, but it has a **strange value"119"**
  - **bmi:** It shows that it may look like Gaussian Distribution
  - **children:** most people have 0 child, number decreses with the children numbers increasing
  - **past claims:** it seems that "<=100000" is the biggest part, past claims have **long tail distribution**, min-max and standardization scaling do not work,it may be replaced the feature by its square root or replace it with its log value.

2.2.2 Use the .value_counts() method to check the categorical data distribution

[5]:
```python
for col in insurance.columns:
    if insurance[col].dtype=='object':
        print()
        print(col)
        print(insurance[col].value_counts())
```

```
sex
male      676
female    662
Name: sex, dtype: int64
```

```
home
South NT            364
North NT            325
Hong Kong Island    325
Kowloon             324
Name: home, dtype: int64

smoker
no     1064
yes     274
Name: smoker, dtype: int64

drinking
occasional    800
frequent      271
no            267
Name: drinking, dtype: int64
```

- **Categorical Data Findings**
  - **sex:** male is a little bit larger than female, but almost evenly distribution
  - **smoker:** non-smoker is larger than smoker
  - **home:** there are four areas, South NT is larger than the other three, but almost evenly distribution
  - **drinking:** most are "occasional drinkers"

3 Visualize and Analyze Data to Gain Insights

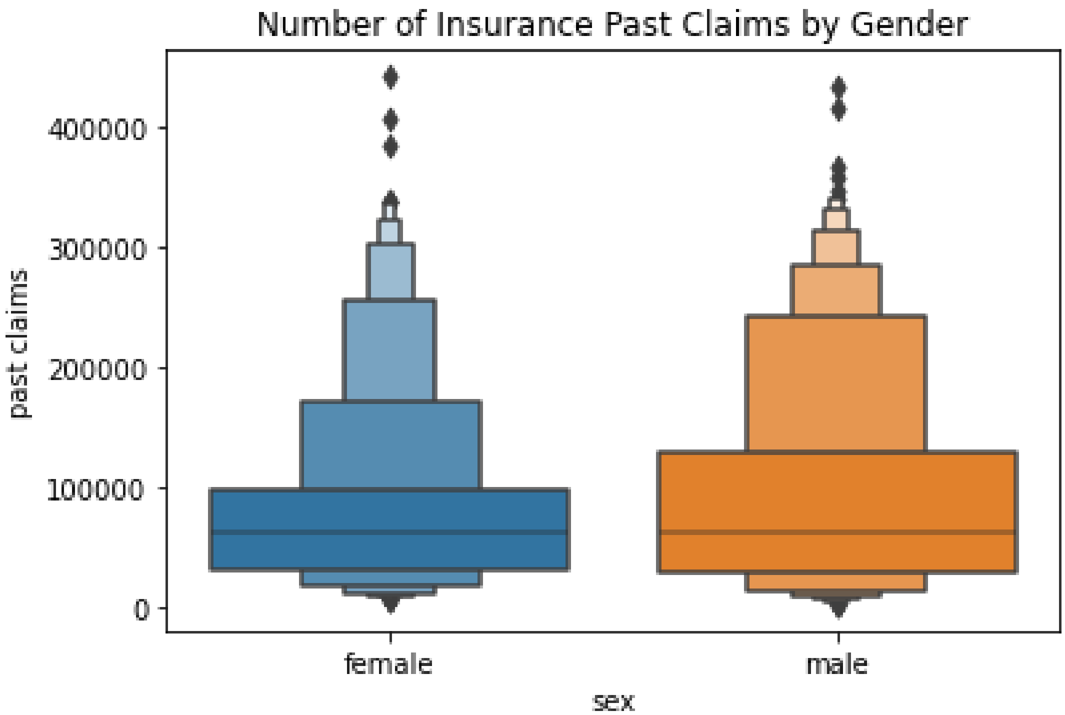
3.1 Analyze the Categorical Variables Relations

3.1.1 Check the Gender with the Past Claims

- Use the boxplot to show the relationship between sex and past claims

[6]:
```
import seaborn as sns
sns.boxenplot(x='sex',y='past claims',data=insurance).set(title='Number of␣
↪Insurance Past Claims by Gender')
```

[6]:
```
[Text(0.5, 1.0, 'Number of Insurance Past Claims by Gender')]
```

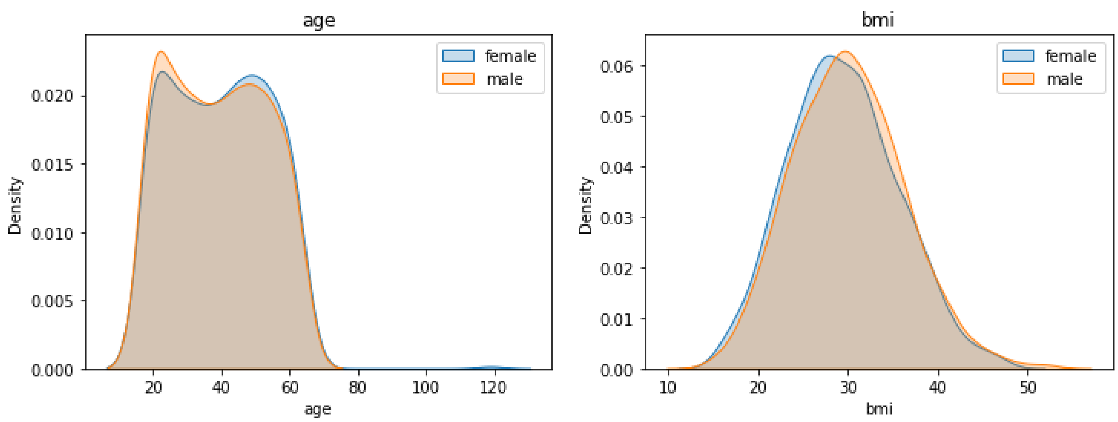


- **Findings**
- Men are more widely distributed compared to women, with more men having higher past claims

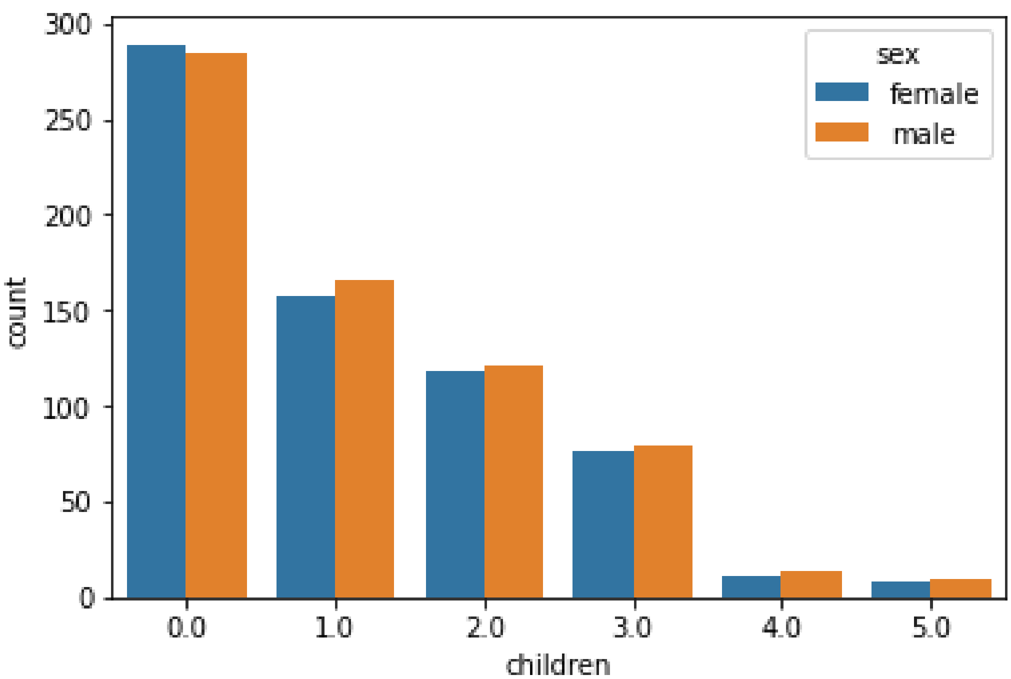
3.1.2 Further check the Gender factors

- Why men have larger past claims than women
- Discover the "bmi","age" deeply into the Gender factors
- Discover the "children" factors
- Discover the "smoker" factors
- Disscover the "drinking" factors

[7]:
```
cols = ['age','bmi']
sex = ['female','male']
fig,ax = plt.subplots(1,2,figsize=(12,4))
for col,p in zip(cols,range(2)):
    for s in sex:
        sns.
 ↪kdeplot(insurance[col][insurance['sex']==s],label=s,ax=ax[p],shade=True)
        ax[p].set_title(col)
        ax[p].legend()
plt.show()
```

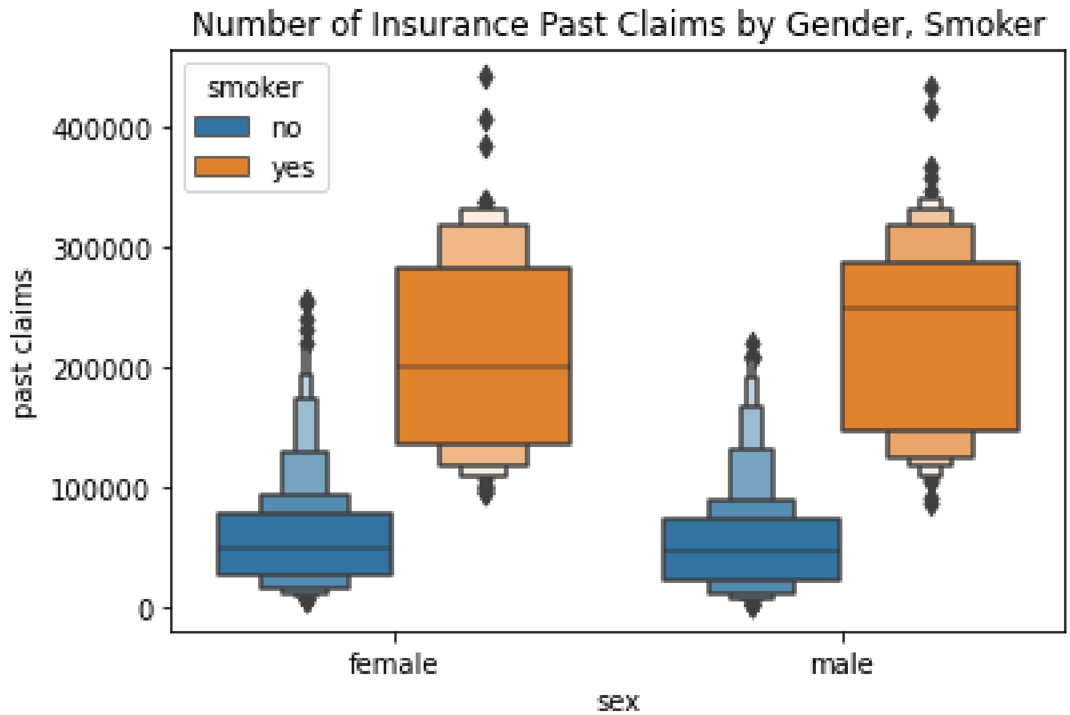


[8]:
```
sns.countplot(data=insurance,x='children',hue='sex')
plt.show()
```



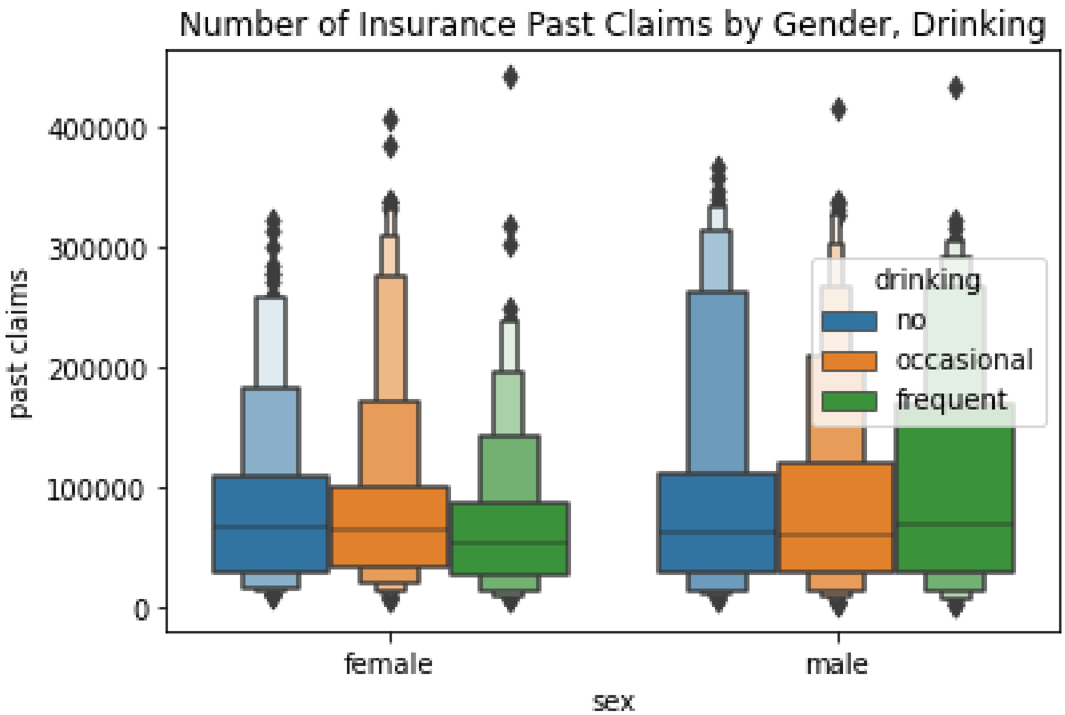
[9]:
```
sns.boxenplot(x='sex',y='past claims',hue='smoker',data=insurance).
 ↪set(title='Number of Insurance Past Claims by Gender, Smoker')
```

[9]: [Text(0.5, 1.0, 'Number of Insurance Past Claims by Gender, Smoker')]



[10]:
```
sns.boxenplot(x='sex',y='past claims',hue='drinking',data=insurance).
    ↪set(title='Number of Insurance Past Claims by Gender, Drinking')
```

[10]: [Text(0.5, 1.0, 'Number of Insurance Past Claims by Gender, Drinking')]

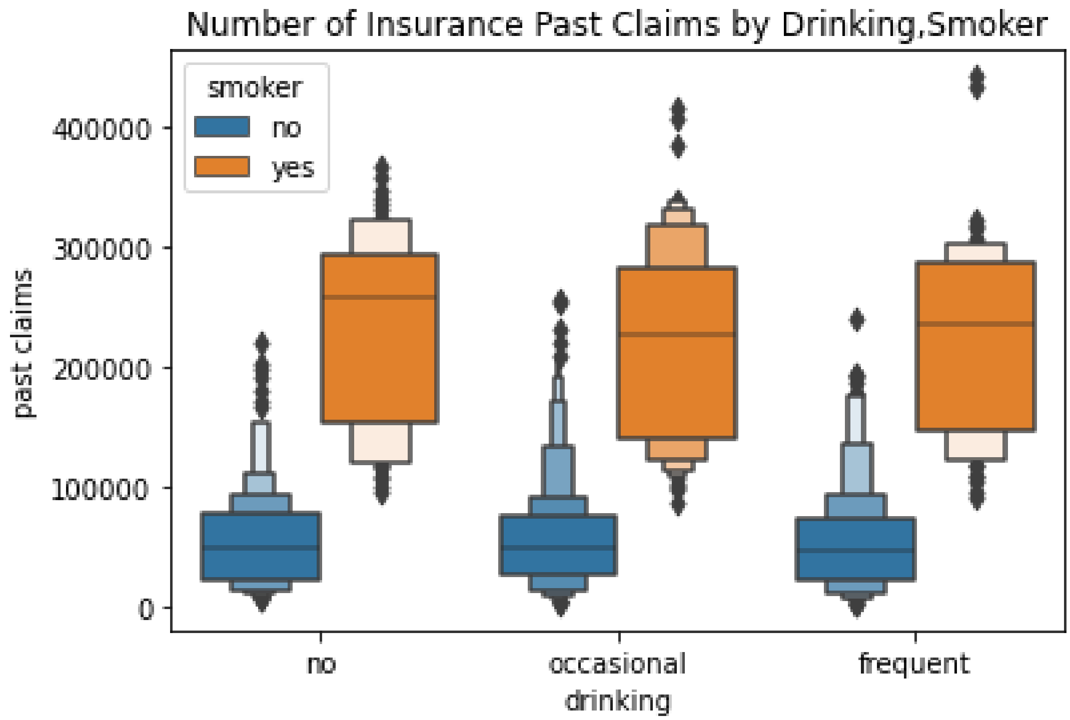


- **Findings**
  - The age and bmi of men and women do not differ much
  - The number of men is larger than women in their 20s and slightly smaller than women in their 50s
  - Men have slightly higher bmi than women
  - However, it shows no significant differences, and it was not age and bmi differences that caused the differences
  - There is little difference between men and women in the number of children they raise
  - It seems that **smoker** has the significant influence in the past claims, female and male's smoker claims are both higher than non-smoker

3.1.3 Discover Smoker, Drinking, Children

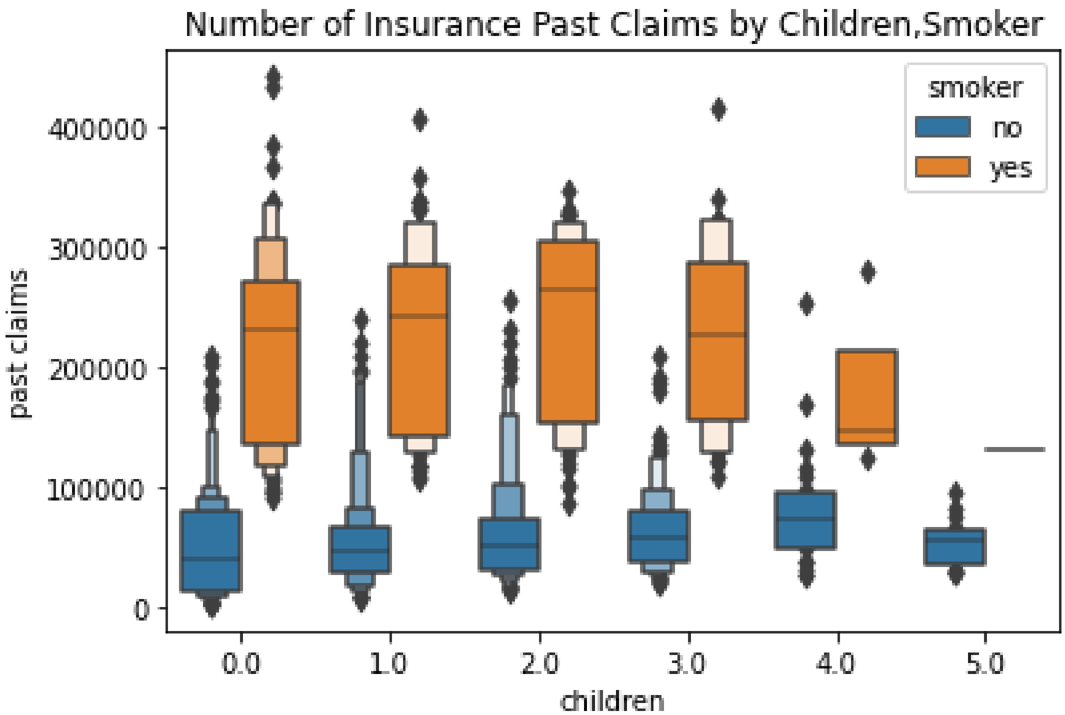
[11]:
```
sns.boxenplot(x='drinking',y='past claims',hue='smoker',data=insurance).
 ↪set(title='Number of Insurance Past Claims by Drinking,Smoker')
```

[11]:
```
[Text(0.5, 1.0, 'Number of Insurance Past Claims by Drinking,Smoker')]
```




```
[12]: sns.boxenplot(x='children',y='past claims',hue='smoker',data=insurance).
       ↪set(title='Number of Insurance Past Claims by Children,Smoker')
```

```
[12]: [Text(0.5, 1.0, 'Number of Insurance Past Claims by Children,Smoker')]
```

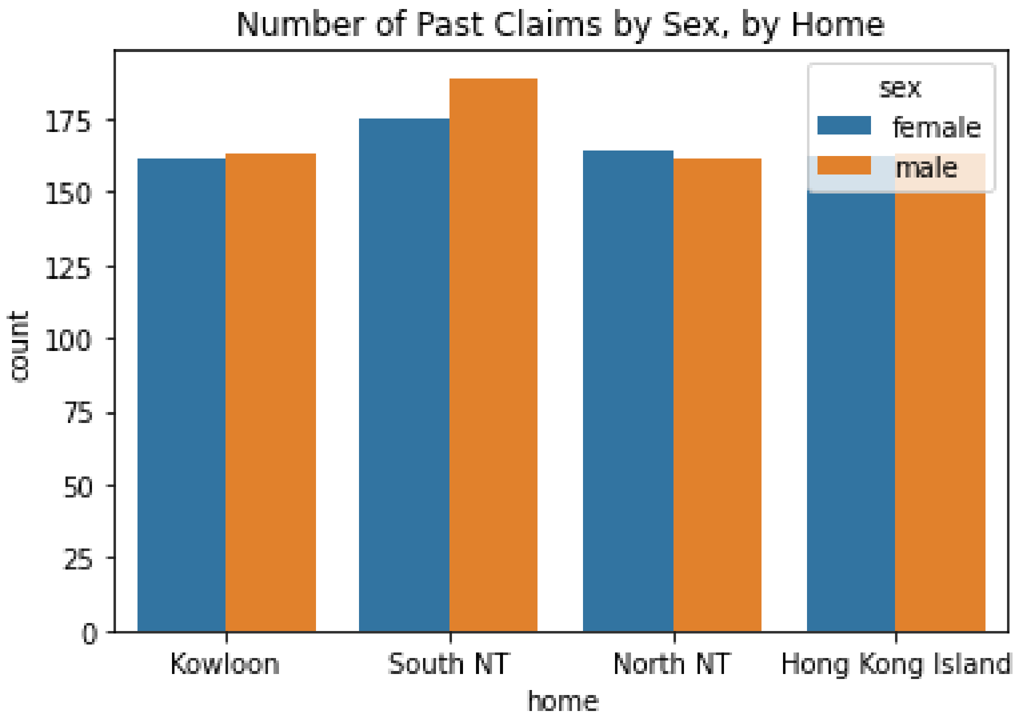


- **Findings**
  - Smoker has higher past claims
  - Different Drinking Degree seems no obvious differences
  - From 0-2 children, with more children more past claims, but over 3 children seems not significant
  - **ALL IN ALL, Smoker has the strong influences on claims**

3.1.4 Check the Home Areas

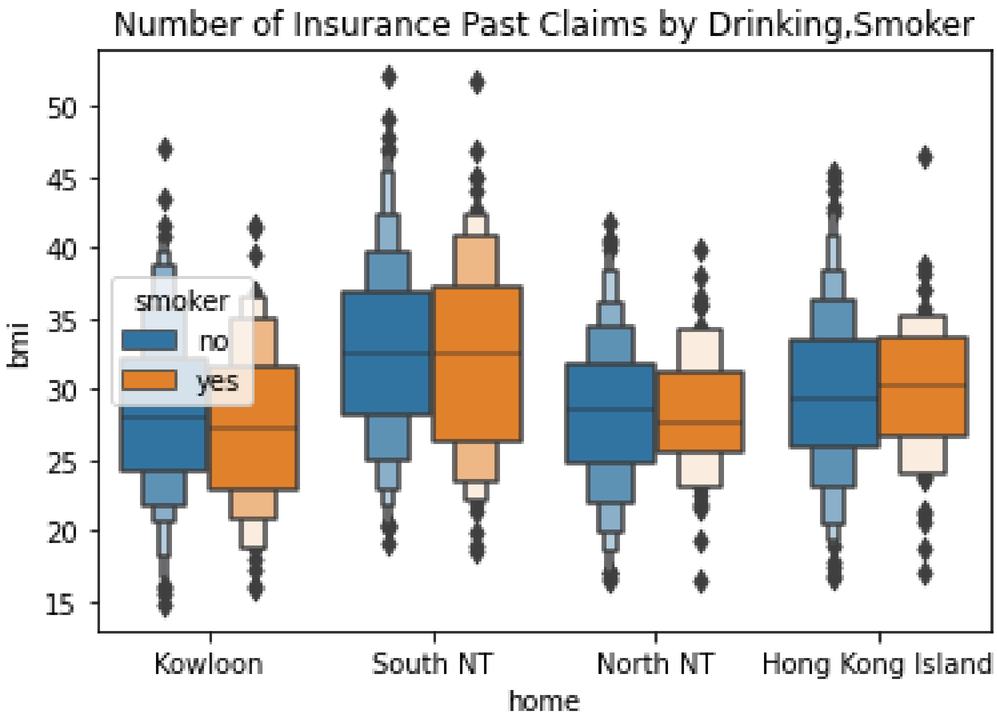
[13]:
```
import seaborn as sns
sns.countplot(x='home',hue='sex',data=insurance).set(title='Number of Past␣
↪Claims by Sex, by Home')
```

[13]:
```
[Text(0.5, 1.0, 'Number of Past Claims by Sex, by Home')]
```



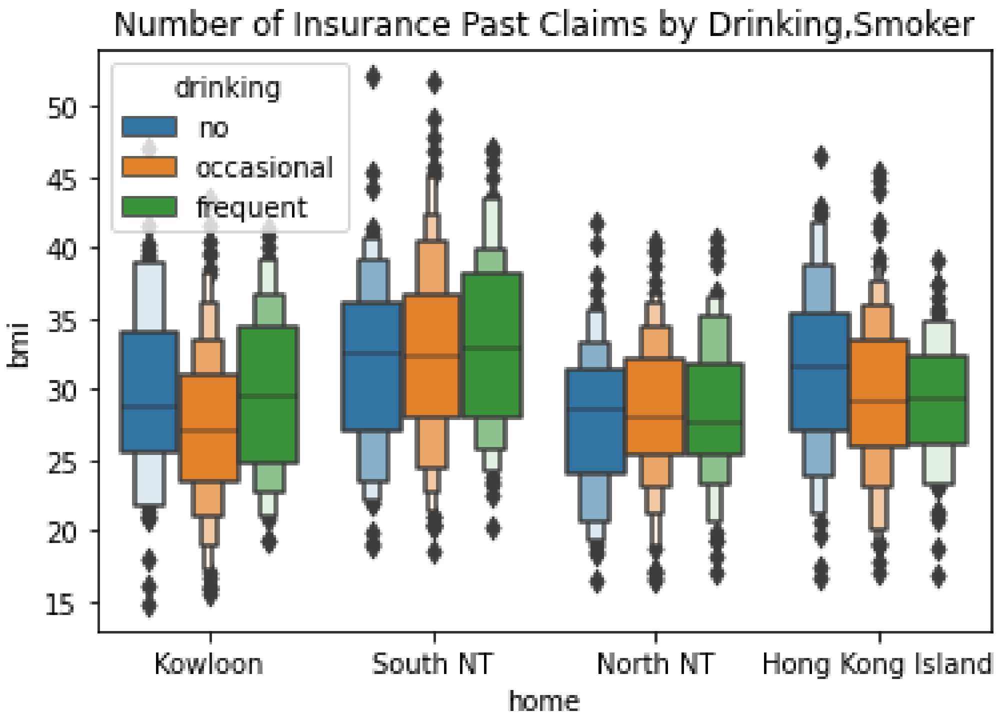
[14]:
```
sns.boxenplot(x='home',y='bmi',hue='smoker',data=insurance).set(title='Number␣
↪of Insurance Past Claims by Drinking,Smoker')
```

[14]:
```
[Text(0.5, 1.0, 'Number of Insurance Past Claims by Drinking,Smoker')]
```



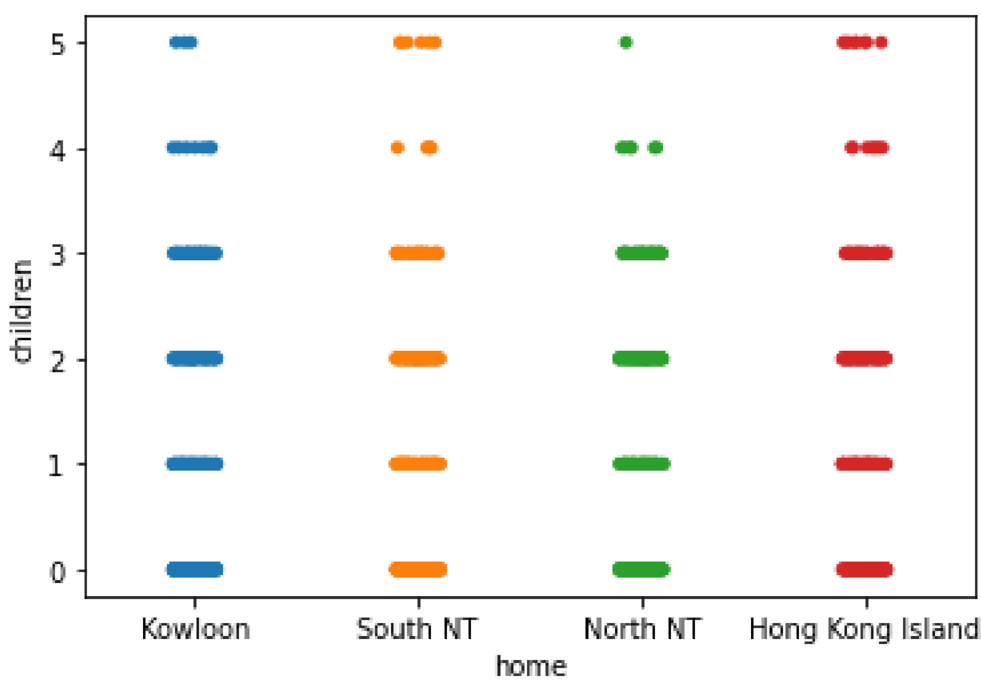
[15]:
```
sns.boxenplot(x='home',y='bmi',hue='drinking',data=insurance).set(title='Number␣
↪of Insurance Past Claims by Drinking,Smoker')
```

[15]:
```
[Text(0.5, 1.0, 'Number of Insurance Past Claims by Drinking,Smoker')]
```



[16]:
```
sns.stripplot(data=insurance,x='home',y='children')
```

[16]: <AxesSubplot:xlabel='home', ylabel='children'>

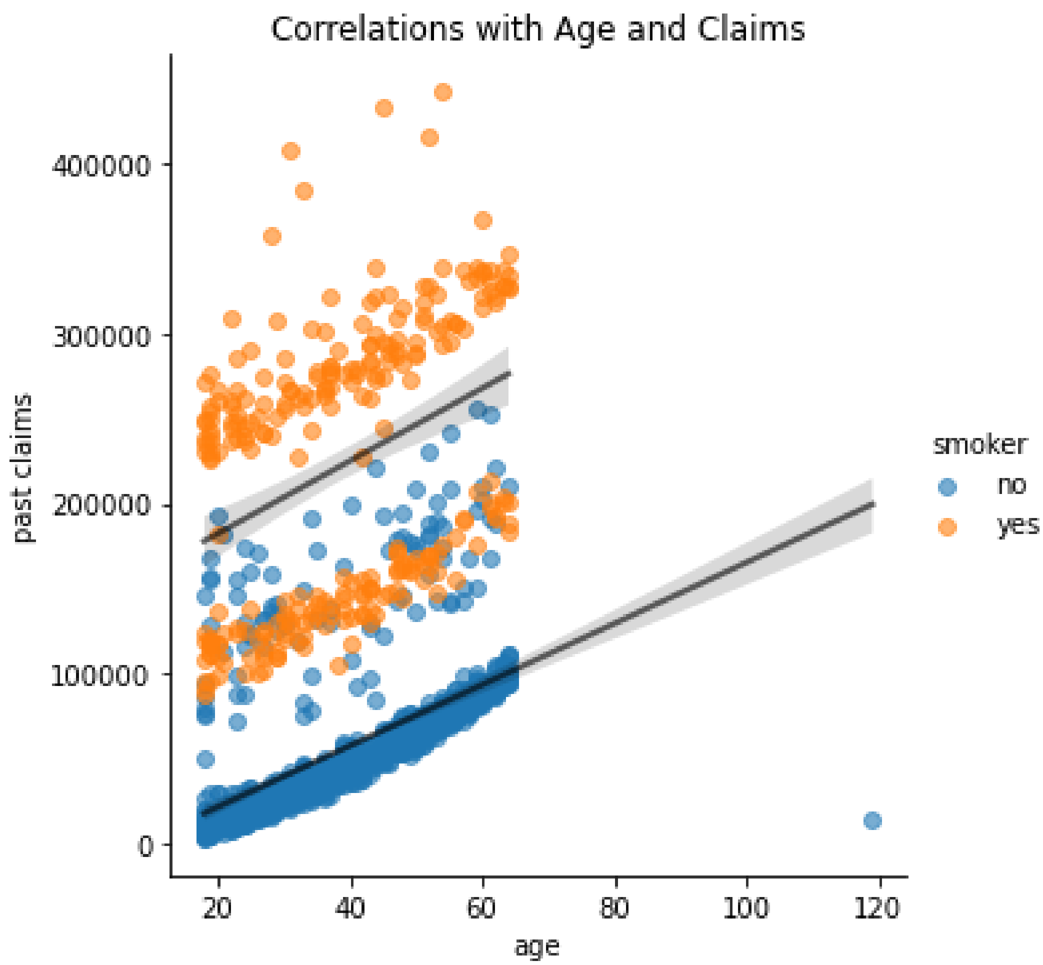


- **Findings**
  - South NT has the highest Past Claims
    - South NT has higher male claims
    - South NT has largest bmi than the other three regions
    - South NT has larger smokers than the other three regions
    - South NT has larger drinking degree than the other three regions
    - Four regions all have similar children distribution
  - we can coclude that South NT has the highest past claims maily because of the men, bmi,smoke, drinking factors

3.1.5 Check the Age and BMI

[17]:
```
# check the age
sns.lmplot(data=insurance,x='age',y='past␣
↪claims',hue='smoker',line_kws=dict(color='black',alpha=0.
↪6),scatter_kws=dict(alpha=0.6)).set(title="Correlations with Age and Claims")
```

[17]:
```
<seaborn.axisgrid.FacetGrid at 0x7faf686213d0>
```

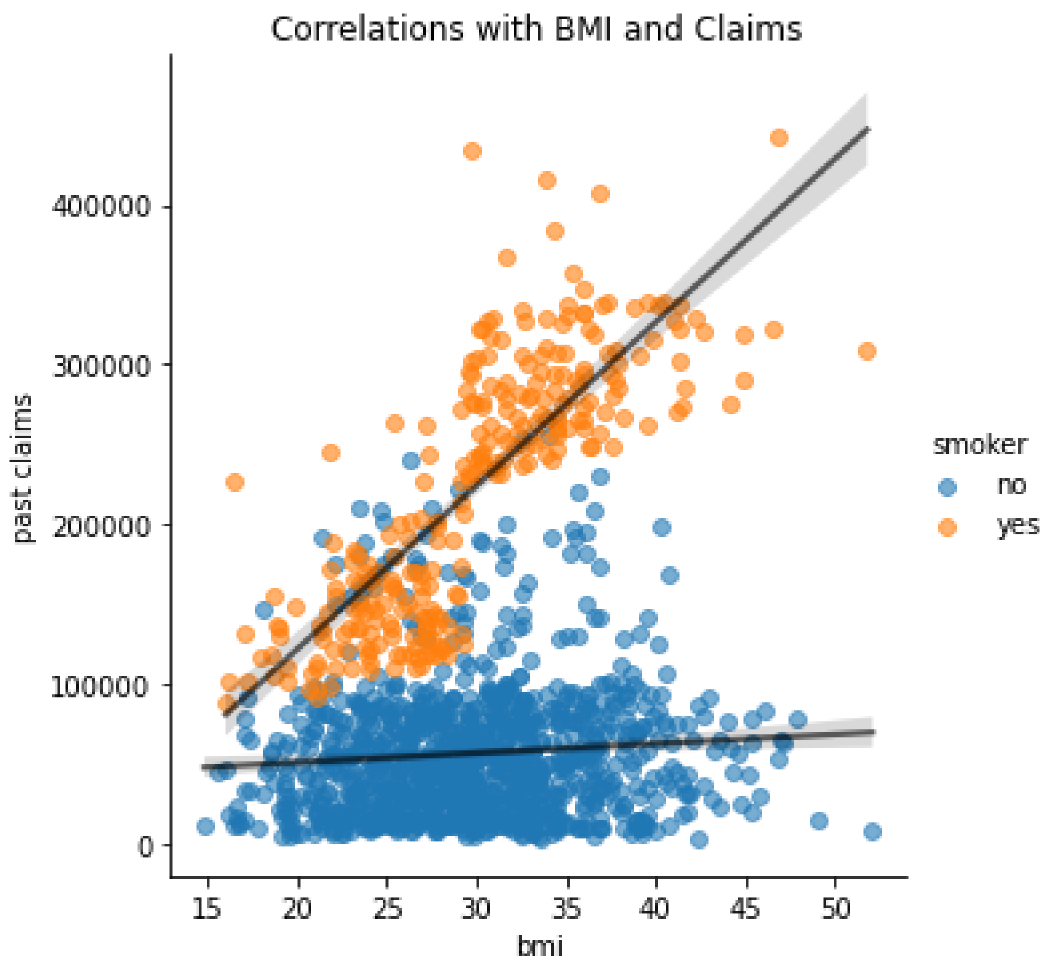


[18]:
```
# check the age
sns.lmplot(data=insurance,x='age',y='past␣
↪claims',hue='drinking',line_kws=dict(color='black',alpha=0.
↪6),scatter_kws=dict(alpha=0.6)).set(title="Correlations with Age and Claims")
```

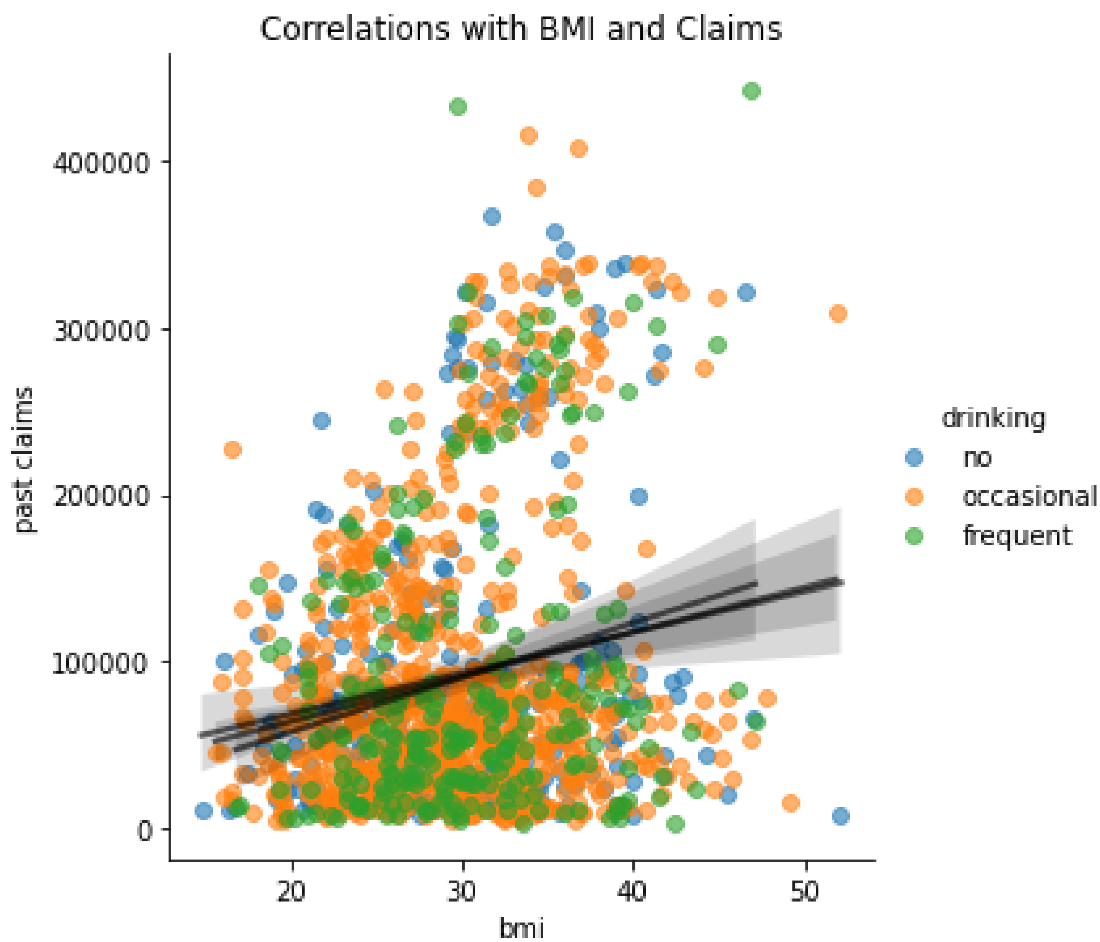
[18]: <seaborn.axisgrid.FacetGrid at 0x7faf48c90eb0>

[19]:

```
# Check the BMI influences
sns.lmplot(data=insurance,x='bmi',y='past␣
↪claims',hue='smoker',line_kws=dict(color='black',alpha=0.
↪6),scatter_kws=dict(alpha=0.6)).set(title="Correlations with BMI and Claims")
plt.show()
```



[20]:
```
# Check the BMI influences
sns.lmplot(data=insurance,x='bmi',y='past␣
↪claims',hue='drinking',line_kws=dict(color='black',alpha=0.
↪6),scatter_kws=dict(alpha=0.6)).set(title="Correlations with BMI and Claims")
plt.show()
```



- **Findings**
  - Age has strong positive relationships with claims
  - Non-smoker past claims increase slowly as BMI increases
  - Smokers past claims increase steeply after bmi exceeds 30(fat benchmark)
  - Drinking has no strong siginificance

4 Prepare the data for Machine Learning algorithms

4.1 Process the Strange Value of Age

[21]:
```python
# delete the over 100 ages, it doesn't meet the real case
insurance.drop(insurance[insurance['age'] >= 100].index, inplace = True)
```

4.2 Process the missing value

[22]:
```
# check the missing value
insurance.isnull()
# finding that data is 1337, but the last 3 rows are all null,so delete the 3␣
↪rows
insurance=insurance.dropna(how='all') #delete the last 3 all null rows
```

[23]:
```
insurance.isnull().any()
# the result is that children column has the null value
#next is to cope with the null value in children column
```

[23]:
```
age             False
sex             False
home            False
bmi             False
children         True
smoker          False
drinking        False
past claims     False
dtype: bool
```

[24]:
```
insurance.isnull().sum()
# check how many null values in the column, 3 values in the children are missing
```

[24]:
```
age             0
sex             0
home            0
bmi             0
children        3
smoker          0
drinking        0
past claims     0
dtype: int64
```

[25]:
```
null_row_idx=insurance.isnull().any(axis=1)
```

[26]:
```
insurance.loc[null_row_idx].head()
```

[26]:
```
       age     sex              home   bmi  children smoker    drinking  \
464   61.0  female  Hong Kong Island  38.4       NaN     no  occasional
1077  22.0    male  Hong Kong Island  34.1       NaN     no  occasional
1317  55.0    male          North NT  31.7       NaN     no          no

      past claims
464       95669.0
1077      20174.0
```

```
1317      73937.0
```

[27]:
```
# choose to fill the median value
median=insurance["children"].median()
insurance["children"].fillna(int(median),inplace=True)
insurance.loc[null_row_idx].head()
```

[27]:
```
       age     sex              home   bmi  children smoker    drinking  \
464   61.0  female  Hong Kong Island  38.4       1.0     no  occasional
1077  22.0    male  Hong Kong Island  34.1       1.0     no  occasional
1317  55.0    male          North NT  31.7       1.0     no          no

      past claims
464       95669.0
1077      20174.0
1317      73937.0
```

4.3 Process the Categorical Data

- sex,home the two definitions have no individual significance, so choose one-hot
- smoker yes,no want to use 0,1 to explain the meaning, so use map method
- drinking has degree, use 0-2 to explain the meaning

[29]:
```
i_data=insurance.copy()
i_data['smoker']=i_data['smoker'].map({'yes':1,'no':0})
i_data['drinking']=i_data['drinking'].map({'no':0,'occasional':1,'frequent':2})
i_data=pd.concat([i_data,pd.get_dummies(i_data['sex']),pd.
 ↪get_dummies(i_data['home'])],axis=1)
del i_data['home']
del i_data['sex']
i_data.head()
```

[29]:
```
    age   bmi  children  smoker  drinking  past claims  female  male  \
0  49.0  32.3       2.0       0         0      72433.0       1     0
1  55.0  29.5       2.0       0         1      79358.0       1     0
2  53.0  26.0       0.0       0         2      67628.0       1     0
3  19.0  33.2       0.0       0         1       7020.0       0     1
4  59.0  36.5       1.0       0         2      82368.0       0     1

   Hong Kong Island  Kowloon  North NT  South NT
0                 0        1         0         0
1                 0        0         0         1
2                 0        0         1         0
3                 1        0         0         0
4                 1        0         0         0
```

4.4 Check the "past claims" feature

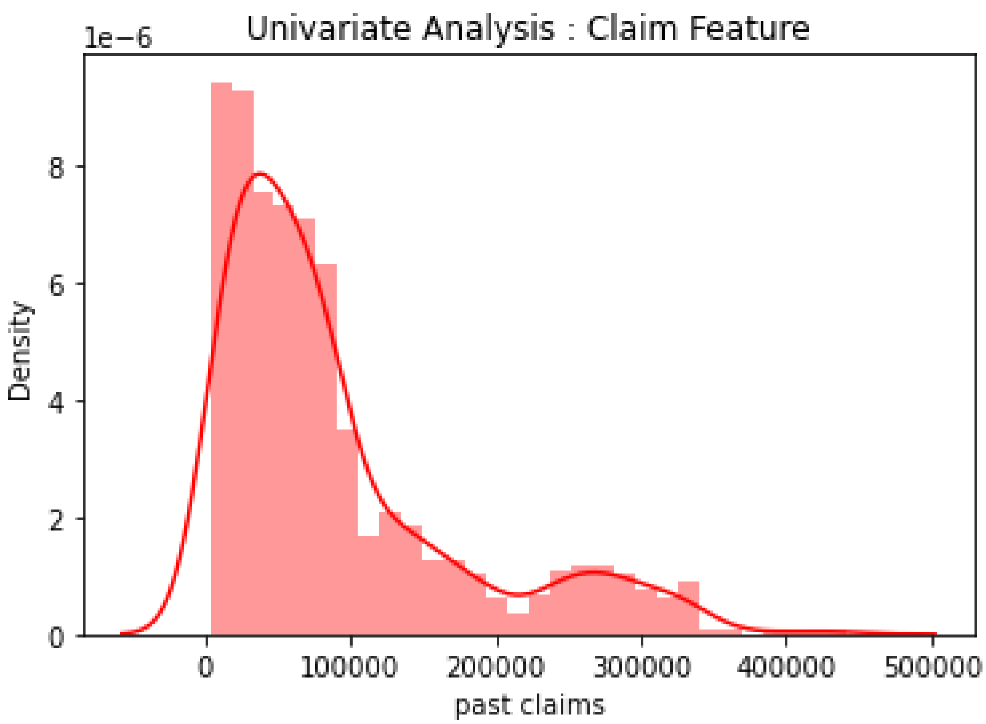
- Before proceeding with machine learning, check the past claims
- I see some outliers and long tail distribution on the "past claims", therefore to do some processes
- but after drop the outliers, it seems not to change significantly, the R square value of the deleted outlier's model even lower
- Therefore, in the final model, I choose not to delete the outliers of the "past claims"

**The distribution of the "past claims"

[30]:
```
sns.distplot(i_data["past claims"], color="r", kde=True).set(title='Univariate␣
↪Analysis : Claim Feature')
```

```
/Users/hebaodan/opt/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

[30]:
```
[Text(0.5, 1.0, 'Univariate Analysis : Claim Feature')]
```

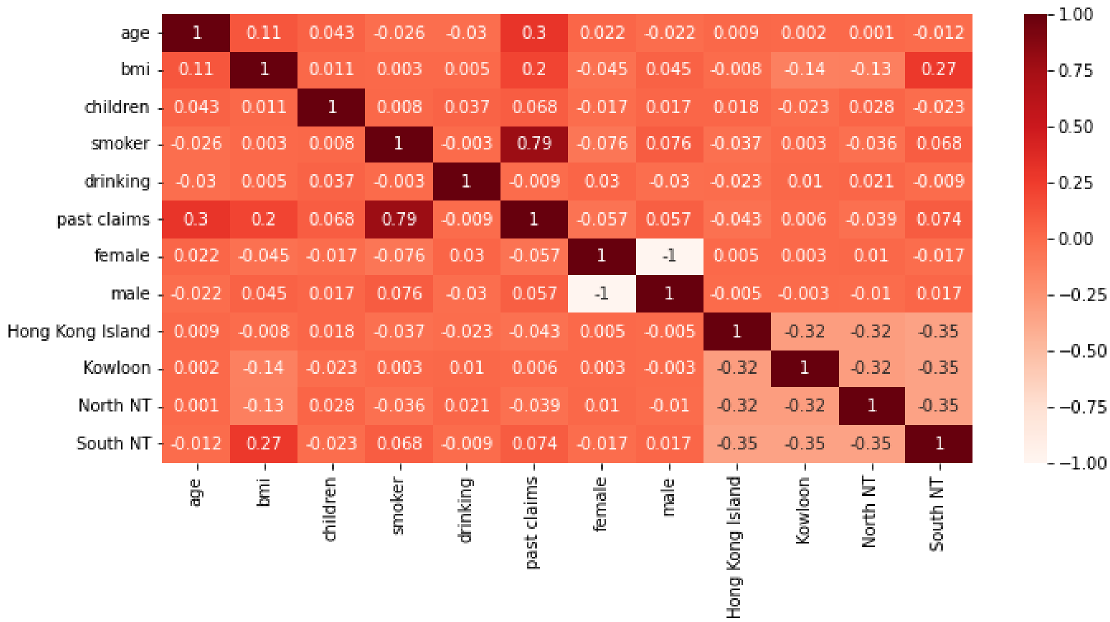


4.5 Check the Correlations

- use the pearson method and draw the heat map to show the correlations

[31]:
```python
corr=i_data.corr(method='pearson').round(3)
plt.figure(figsize=(11,5))
sns.heatmap(corr,cmap='Reds',annot=True)
```

[31]:
```
<AxesSubplot:>
```



- **Findings**
  - in the past claims row, we see that age is the largest, next is smoker, bmi
  - the result is same as the before visualization result

5 Data Modeling

5.1 Define the Dependent Variables in X and Independent Variable in Y

[32]:
```python
X=i_data.drop(['past claims'],axis=1)
y=i_data.loc[:,'past claims']
```

5.2 Standardize the data values to avoid biased outcome

[33]:
```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled=scaler.fit_transform(X)
```

5.3 Split the data

[34]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_scaled,y, test_size=0.2,␣
↪random_state=42)
print("Train", X_train.shape, "and test", X_test.shape)
```

```
Train (1069, 11) and test (268, 11)
```

5.4 Training model with training set

[35]:
```
from sklearn.linear_model import LinearRegression

lin_reg =LinearRegression()

model=lin_reg.fit(X_train, y_train)
```

[36]:
```
model.score(X_test,y_test)
```

[36]:
```
0.7336686079691823
```

5.5 Use test data to predict and evaluate the model

[37]:
```
## create function to fit models
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
model_preds = []
import numpy as np

def fit_model(model, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    r2 = round(r2_score(y_test, y_pred),4)
    adj_r2 = round(1 - (1-r2)*(len(y)-1)/(len(y)-X.shape[1]-1),4)
    mse = round(mean_squared_error(y_test, y_pred),4)
    mae = round(mean_absolute_error(y_test, y_pred),4)
    rmse = round(np.sqrt(mean_squared_error(y_test, y_pred)),4)
    model_preds.append([model_name, r2, adj_r2, mse, mae, rmse])
    print ("The R-Squared Value is: ", r2)
    print ("Adjusted R-Squared Value is: ", adj_r2)
    print("The Mean Squared error (MSE) is: ", mse)
    print("Root Mean Squared Error (RMSE): ", rmse)
    print("Mean Absolute Error (MAE) is: ", mae)

## model evaluation function
def model_eval():
    preds = pd.DataFrame(model_preds)
    preds.columns = ["Mod_Name", "R2 Value", "adj_R2", "MSE", "RMSE", "MAE"]
```

```
    return preds.sort_values(by="R2 Value", ascending=False)
```

[38]:
```
fit_model(model, "Linear Regression")
```

```
The R-Squared Value is:  0.7337
Adjusted R-Squared Value is:  0.7315
The Mean Squared error (MSE) is:  1797765801.8695
Root Mean Squared Error (RMSE):  42400.0684
Mean Absolute Error (MAE) is:  29629.7618
```

**Findings of the model - it seems the accuary of the Linear Regression is not so high - For higher accuracy, it is suggested to train other models, like DecisionTree, RandomForest, K-Neighbors Regression

[39]:
```
print(model.intercept_.round(),model.coef_.round())
```

```
90671.0 [ 2.60110000e+04  1.46180000e+04  4.64100000e+03  6.74380000e+04
  2.53000000e+02 -1.40527477e+14 -1.40527477e+14 -6.10050106e+16
 -6.09411714e+16 -6.09411714e+16 -6.33053890e+16]
```

5.6 Conclusion and Insights

- As the age of each year increases, assuming that everything is the same (unchanged), we will expect an average of 2.6 medical expenses.
- For each unit of BMI, the annual medical expenses will increase an average of 1.46.
- Similarly, each child adds an average of 4.64 additional medical expenses each year;
- Smoking people spend far more than non -smokers, the higher the degree of Drinking, the higher the cost
- SOUTH NT tends to have the highest average medical expenses.
- **The result in the linear regression model is logical**
- Smoking, smoking and obesity are often linked to other health issues, and additional family members or recipients may lead to increased number of diagnosis and increase in prevention of health care costs, resulting in increased costs

[40]:
```
#Draw the Fitted image
y_pre = model.predict(X_test)
sns.scatterplot(y_test,y_pre,label='LinearRegression')
sns.lineplot(x=[0,50000],y=[0,50000],label='perfect',color='black')
plt.legend()
# the result shows that at the first stage The fitting effect of linear␣
 ↪regression is good, but the fitting effect is not good after the past claims␣
 ↪become larger
```

```
/Users/hebaodan/opt/anaconda3/lib/python3.9/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
```

```
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

[40]: <matplotlib.legend.Legend at 0x7faf4b209c70>