r-regression-for-salary-prediction

July 13, 2023

[1]:
```
# linear regression on salary prediction
import pandas as pd   # for working with salary dataset
import numpy as np    # for working with arrays
import matplotlib.pyplot as plt # data visualization
import seaborn as sns          # data visualization
```

[4]:
```
data = pd.read_csv('Salary.csv')
data.head()
```

[4]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

```
<google.colab._quickchart_helpers.SectionTitle at 0x796797c514e0>
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_3642660912639395343, *['YearsExperience'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')
```

```
def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_3642660912639395343, *['Salary'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x796794c1e1d0>
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_3642660912639395343, *['YearsExperience'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_3642660912639395343, *['Salary'], **{})
```

```
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7967948ac640>
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')

def scatter_plots(df, colname_pairs, scatter_plot_size=2.5, size=8, alpha=.6):
  from matplotlib import pyplot as plt
  plt.figure(figsize=(len(colname_pairs) * scatter_plot_size, scatter_plot_size))
  for plot_i, (x_colname, y_colname) in enumerate(colname_pairs, start=1):
    ax = plt.subplot(1, len(colname_pairs), plot_i)
    ax.scatter(df[x_colname], df[y_colname], s=size, alpha=alpha)
    plt.xlabel(x_colname)
    plt.ylabel(y_colname)
    ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = scatter_plots(df_3642660912639395343, *[[['YearsExperience',␣
 ↪'Salary']]], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7967949a3f10>
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')

def time_series_multiline(df, timelike_colname, value_colname, series_colname,␣
 ↪figsize=(2.5, 1.3), mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index %␣
 ↪len(palette)])
```

```
  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(df_3642660912639395343, *['YearsExperience',␣
 ↪'Salary', None], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_3642660912639395343 = autoviz.get_registered_df('df_3642660912639395343')

def time_series_multiline(df, timelike_colname, value_colname, series_colname,␣
 ↪figsize=(2.5, 1.3), mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index %␣
 ↪len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
```

```
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(df_3642660912639395343, *['YearsExperience',␣
↪'count()', None], **{})
chart
```

```
Error: Runtime no longer has a reference to this dataframe, please re-run this
cell and try again.
```

The dataset value as observed above is continuous values. Hence, applying regression analysis.

[5]:
```
data.tail()
```

[5]:

	YearsExperience	Salary
30	11.2	127345
31	11.5	126756
32	12.3	128765
33	12.9	135675
34	13.5	139465

[6]:
```
data.columns # returns all the columns ie 2 here with the columns as object type
```

[6]:
```
Index(['YearsExperience', 'Salary'], dtype='object')
```

[7]:
```
data.shape  # dimensions shows (rows x columns) total 35 rows here and 2 columns
```

[7]:
```
(35, 2)
```

[8]:
```
data.describe()
```

[8]:

	YearsExperience	Salary
count	35.000000	35.000000
mean	6.308571	83945.600000
std	3.618610	32162.673003
min	1.100000	37731.000000
25%	3.450000	57019.000000
50%	5.300000	81363.000000
75%	9.250000	113223.500000
max	13.500000	139465.000000

```
<google.colab._quickchart_helpers.SectionTitle at 0x796797ae0640>
```

```
import numpy as np
from google.colab import autoviz
```

```
df_3751262681771091102 = autoviz.get_registered_df('df_3751262681771091102')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_3751262681771091102, *['YearsExperience'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_3751262681771091102 = autoviz.get_registered_df('df_3751262681771091102')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_3751262681771091102, *['Salary'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7967979a1f00>
```

```
import numpy as np
from google.colab import autoviz
df_3751262681771091102 = autoviz.get_registered_df('df_3751262681771091102')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()
```

```
chart = histogram(df_3751262681771091102, *['YearsExperience'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_3751262681771091102 = autoviz.get_registered_df('df_3751262681771091102')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_3751262681771091102, *['Salary'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x796795284b80>
```

```
import numpy as np
from google.colab import autoviz
df_3751262681771091102 = autoviz.get_registered_df('df_3751262681771091102')

def scatter_plots(df, colname_pairs, scatter_plot_size=2.5, size=8, alpha=.6):
  from matplotlib import pyplot as plt
  plt.figure(figsize=(len(colname_pairs) * scatter_plot_size, scatter_plot_size))
  for plot_i, (x_colname, y_colname) in enumerate(colname_pairs, start=1):
    ax = plt.subplot(1, len(colname_pairs), plot_i)
    ax.scatter(df[x_colname], df[y_colname], s=size, alpha=alpha)
    plt.xlabel(x_colname)
    plt.ylabel(y_colname)
    ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = scatter_plots(df_3751262681771091102, *[[['YearsExperience',␣
 ↪'Salary']]], **{})
chart
```

[10]:
```
# blank values and NAN are considered as missing values which can be replaced␣
 ↪by mean or medians
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
```

```
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  35 non-null     float64
 1   Salary           35 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 688.0 bytes
```

[11]:
```
# checking for null values
data.isnull()
```

[11]:

	YearsExperience	Salary
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False
25	False	False
26	False	False
27	False	False
28	False	False
29	False	False
30	False	False
31	False	False
32	False	False
33	False	False
34	False	False

```
No charts were generated by quickchart
```

[12]:
```
# to show total of null values
data.isnull().any()
```

[12]:
```
YearsExperience    False
Salary             False
dtype: bool
```

As observed above both the columns result false value which indicates that there are no null values

[13]:
```
# summation of the null value
data.isnull().sum()
```

[13]:
```
YearsExperience    0
Salary             0
dtype: int64
```

DATA VISUALIZATION

[22]:
```
plt.plot(data['Salary'], color='limegreen')
plt.xlabel("Salary")
plt.title("Employee Salary")
```

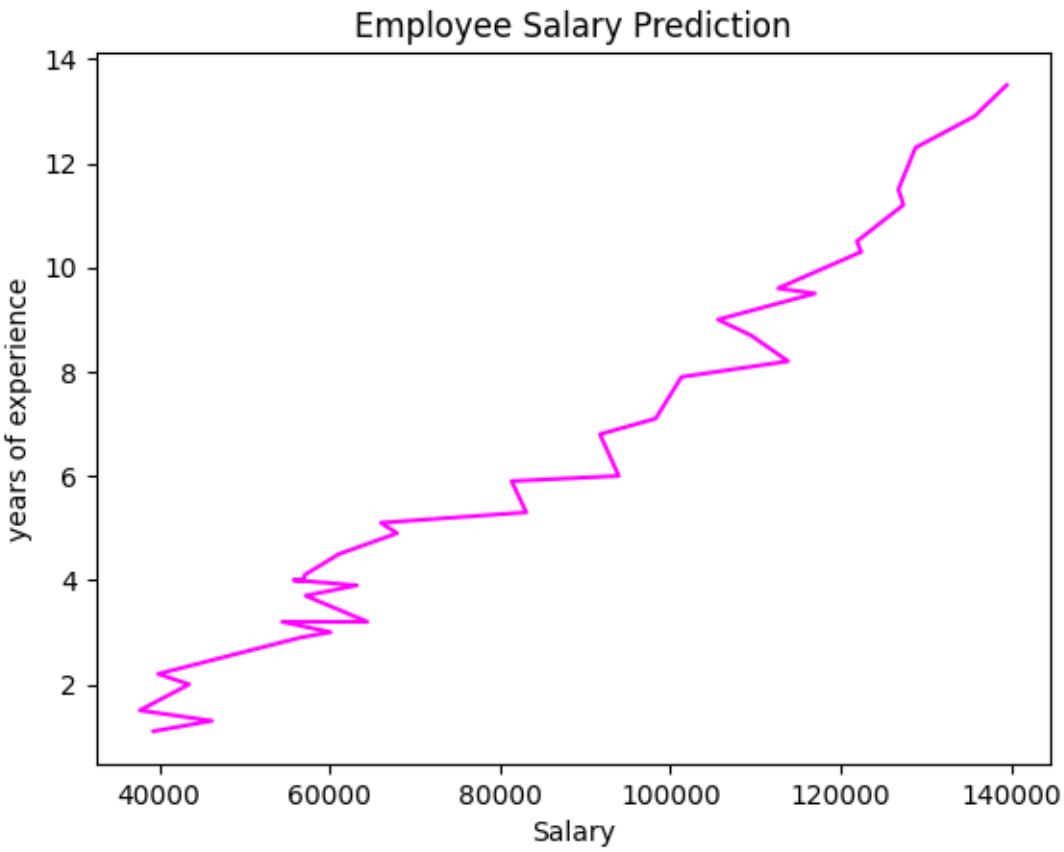
[22]:
```
Text(0.5, 1.0, 'Employee Salary')
```

[23]:
```
plt.plot(data['YearsExperience'], color='blueviolet')
plt.xlabel("YearsExperience")
plt.title("Employee's Experience")
```

[23]: Text(0.5, 1.0, "Employee's Experience")

[24]:
```
# visualizing the two columns
plt.plot(data['Salary'],data['YearsExperience'], color='magenta')
plt.xlabel("Salary")
plt.ylabel("years of experience")
plt.title("Employee Salary Prediction")
```

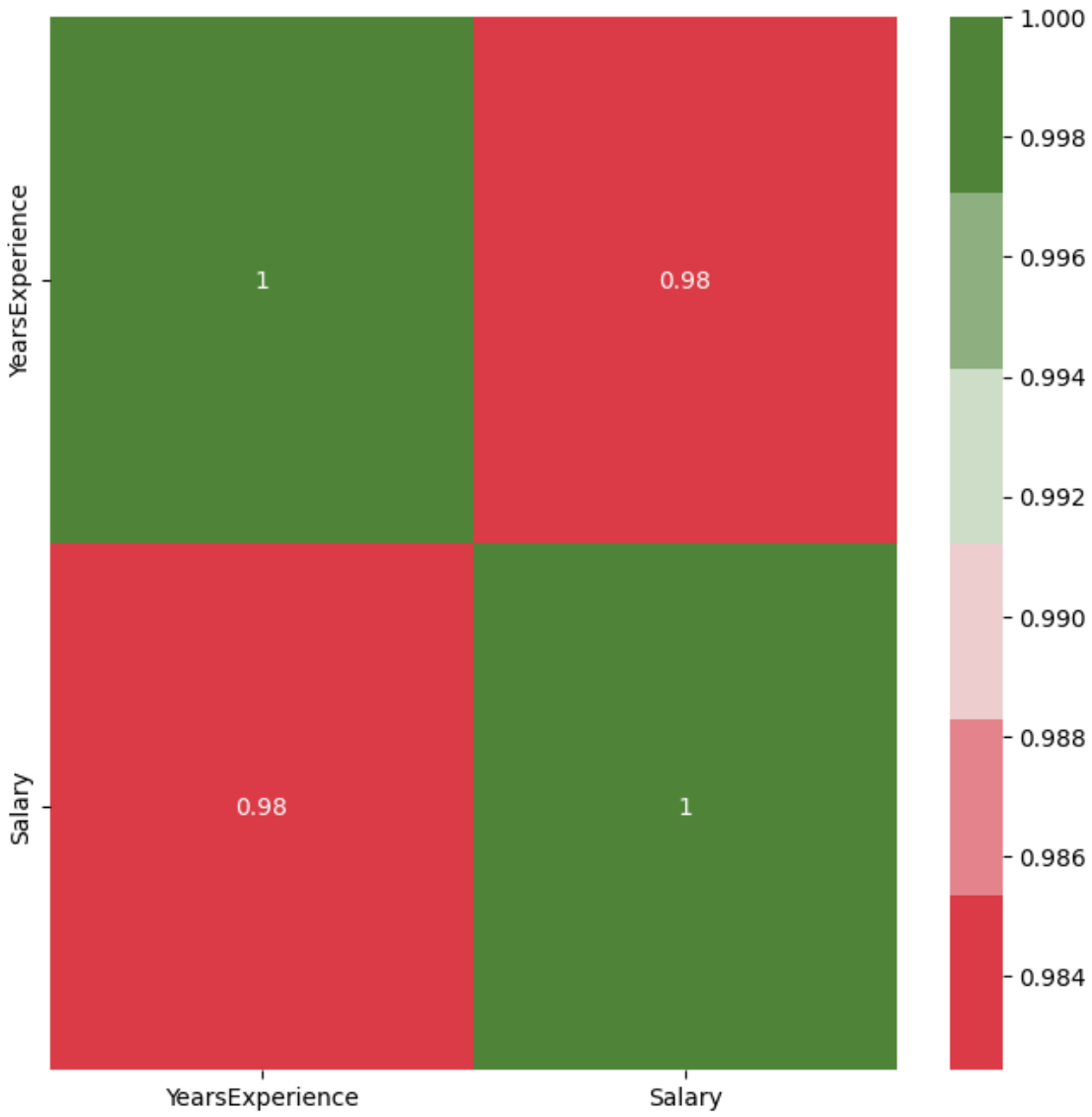
[24]: Text(0.5, 1.0, 'Employee Salary Prediction')



[61]:
```
f , ax = plt.subplots(figsize=(8,8))
corr = data.corr("pearson")

sns.heatmap(corr, ax=ax    # ax=Axes in which to draw the plot
            , annot=True   # writes the value of the data in each set if set to␣
 ↪TRUE
            , cmap=sns.diverging_palette(10,120) # matplotlib color map
            )
```

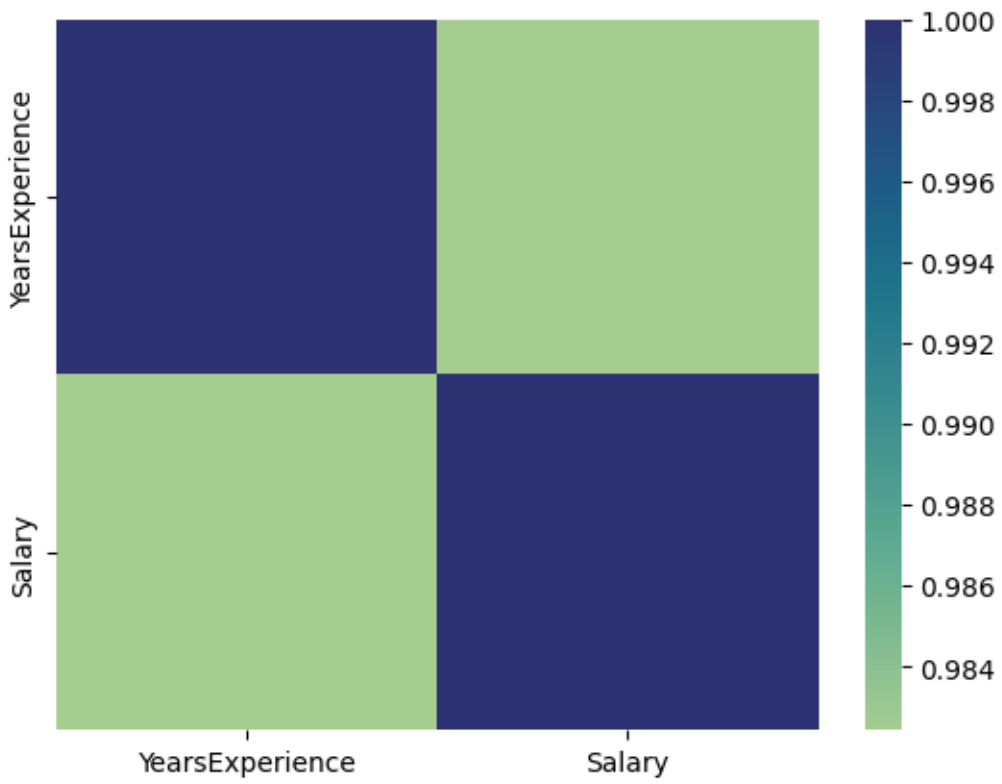
[61]: <Axes: >



OBSERVING THE HEATMAPS IN DIFFERENT COLOR RANGES

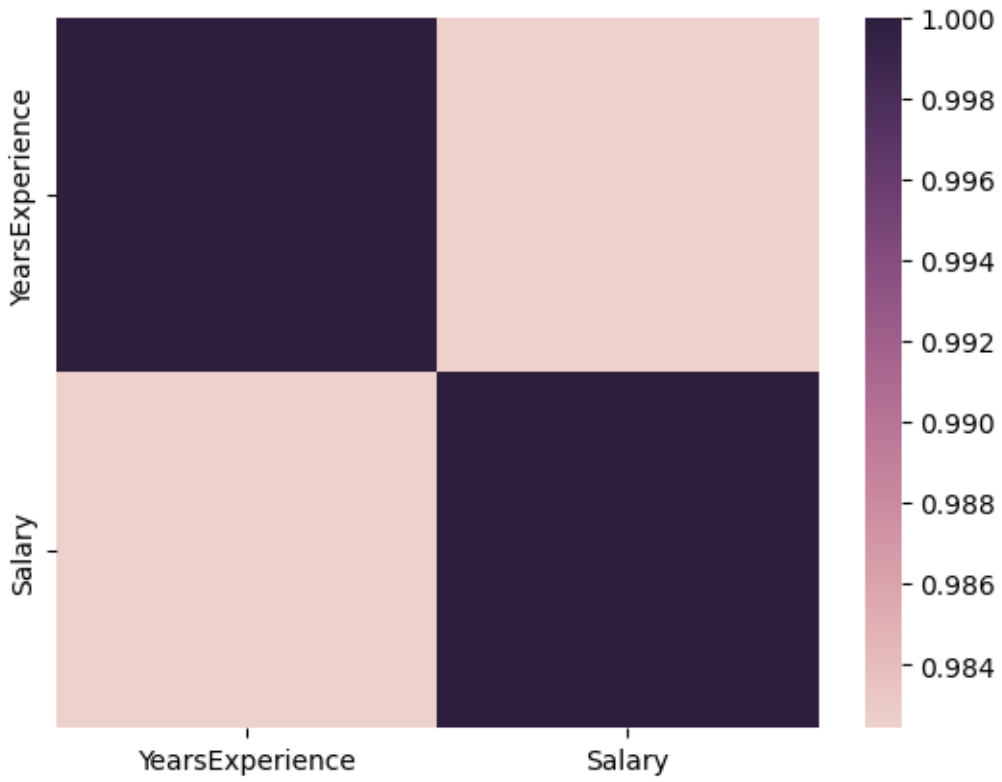
[63]:
```
sns.heatmap(corr, cmap="crest")
```

[63]: <Axes: >



[64]:
```
sns.heatmap(corr, cmap=sns.cubehelix_palette(as_cmap=True))
```

[64]: <Axes: >



MODEL TRAINING AND TESTING

[44]:
```
from sklearn.model_selection import train_test_split
```

[45]:
```
x = data.drop("Salary" , axis = 1)  # dropping salary columns and storing it to␣
↪x
```

[47]:
```
y = data['Salary']
y.head()
```

[47]:
```
0    39343
1    46205
2    37731
3    43525
4    39891
Name: Salary, dtype: int64
```

[48]:
```
# printing x and y
print(x.head())
print(y.head())
```

```
   YearsExperience
0              1.1
```

```
1              1.3
2              1.5
3              2.0
4              2.2
0    39343
1    46205
2    37731
3    43525
4    39891
Name: Salary, dtype: int64
```

It is observed that x is now the years of experience and y is the salary

[53]:
```
xtrain , xtest , ytrain , ytest = train_test_split (x , y ,test_size = 0.2,␣
 ↪random_state = 42)
# training 80 % of dataset and 20% for testing as given in test size
# also random state set to 42 describes that the data can go in any random␣
 ↪order for training and testing

from sklearn.linear_model import LinearRegression
L = LinearRegression()
```

[54]:
```
L.fit(xtrain , ytrain)
```

[54]:
```
LinearRegression()
```

[56]:
```
y_pred = L.predict(xtest)
print(L.score(xtest , ytest))
```

```
0.8914234140042779
```

The score for the predicted value above shows that 89% accurately the model is working.

[59]:
```
print("actual salary" , y)
```

```
actual salary 0      39343
1      46205
2      37731
3      43525
4      39891
5      56642
6      60150
7      54445
8      64445
9      57189
10     63218
11     55794
12     56957
13     57081
```

```
14     61111
15     67938
16     66029
17     83088
18     81363
19     93940
20     91738
21     98273
22    101302
23    113812
24    109431
25    105582
26    116969
27    112635
28    122391
29    121872
30    127345
31    126756
32    128765
33    135675
34    139465
Name: Salary, dtype: int64
```

[58]:
```
print("predicted salary" , y_pred)
```

```
predicted salary [110576.91706292  64251.57268882 103713.90308157
89987.87511888
  71114.58667017 119155.68453961  80551.23089452]
```