Salary Prediction - Exploratory Data Analysis

```
The script loads, explores, and visualizes the salary prediction datasets.

Author = Marzena Porebski
Email = mmazur5@yahoo.com
```

In [130]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Load the data

In [131]:
```
#Read in files into a Pandas dataframe
Job_Data = pd.read_csv('E:/job_data.csv')
Salaries = pd.read_csv('E:/salaries.csv')
```

Examine the data

In [132]:
```
Job_Data.head(5)
```

Out[132]:

	jobId	companyId	jobType	degree	major	industry	yearsExperience	m
0	JOB1362684407687	COMP37	CFO	MASTERS	MATH	HEALTH	10	
1	JOB1362684407688	COMP19	CEO	HIGH_SCHOOL	NONE	WEB	3	
2	JOB1362684407689	COMP52	VICE_PRESIDENT	DOCTORAL	PHYSICS	HEALTH	10	
3	JOB1362684407690	COMP38	MANAGER	DOCTORAL	CHEMISTRY	AUTO	8	
4	JOB1362684407691	COMP7	VICE_PRESIDENT	BACHELORS	PHYSICS	FINANCE	8	

In [133]:
```
Salaries.head(5)
```

Out[133]:

	jobId	salary
0	JOB1362684407687	130
1	JOB1362684407688	101
2	JOB1362684407689	137
3	JOB1362684407690	142
4	JOB1362684407691	163

In [134]:
```
Job_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 8 columns):
jobId                  1000000 non-null object
companyId              1000000 non-null object
jobType                1000000 non-null object
degree                 1000000 non-null object
major                  1000000 non-null object
industry               1000000 non-null object
yearsExperience        1000000 non-null int64
milesFromMetropolis    1000000 non-null int64
dtypes: int64(2), object(6)
memory usage: 61.0+ MB
```

In [135]:
```
Salaries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 2 columns):
jobId     1000000 non-null object
salary    1000000 non-null int64
dtypes: int64(1), object(1)
memory usage: 15.3+ MB
```

Clean the data

look for duplicate data, invalid data (e.g. salaries <=0), or corrupt data and remove it

In [136]:
```
Job_Data.duplicated().sum()
```

Out[136]:
```
0
```

In [137]:
```
Salaries.duplicated().sum()
```

Out[137]:
```
0
```

Identify numerical and categorical variables

In [138]:
```
Job_Data.columns
```

Out[138]:
```
Index(['jobId', 'companyId', 'jobType', 'degree', 'major', 'industry',
       'yearsExperience', 'milesFromMetropolis'],
      dtype='object')
```

In [139]:
```
numeric_cols = ['yearsExperience', 'milesFromMetropolis']
```

In [140]:
```
categorical_cols = ['jobId', 'companyId', 'jobType', 'degree', 'major', 'industry']
```

Summarize numerical and categorical variables separately

In [141]:
```
Job_Data.describe(include = [np.number])
```

Out[141]:

	yearsExperience	milesFromMetropolis
count	1000000.000000	1000000.000000
mean	11.992386	49.529260
std	7.212391	28.877733
min	0.000000	0.000000
25%	6.000000	25.000000
50%	12.000000	50.000000
75%	18.000000	75.000000
max	24.000000	99.000000

In [142]:
```
Job_Data.describe(include = ['O'])
```

Out[142]:

	jobId	companyId	jobType	degree	major	industry
count	1000000	1000000	1000000	1000000	1000000	1000000
unique	1000000	63	8	5	9	7
top	JOB1362684508638	COMP39	SENIOR	HIGH_SCHOOL	NONE	WEB
freq	1	16193	125886	236976	532355	143206

Merge two tables into single data table

In [143]:
```
Salary_Predictions = pd.merge(Job_Data, Salaries, on='jobId')
```

In [144]:
```
Salary_Predictions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000000 entries, 0 to 999999
Data columns (total 9 columns):
jobId                  1000000 non-null object
companyId              1000000 non-null object
jobType                1000000 non-null object
degree                 1000000 non-null object
major                  1000000 non-null object
industry               1000000 non-null object
yearsExperience        1000000 non-null int64
milesFromMetropolis    1000000 non-null int64
salary                 1000000 non-null int64
dtypes: int64(3), object(6)
memory usage: 76.3+ MB
```

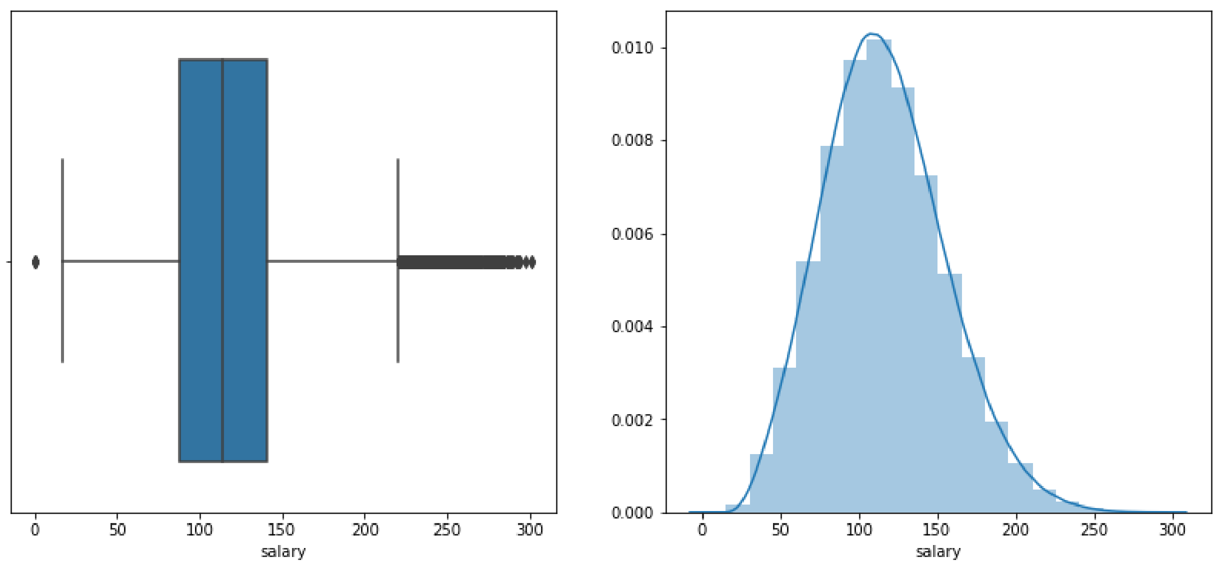
In [145]:
```
Salary_Predictions.head()
```

Out[145]:

	jobId	companyId	jobType	degree	major	industry	yearsExperience	m
0	JOB1362684407687	COMP37	CFO	MASTERS	MATH	HEALTH	10	
1	JOB1362684407688	COMP19	CEO	HIGH_SCHOOL	NONE	WEB	3	
2	JOB1362684407689	COMP52	VICE_PRESIDENT	DOCTORAL	PHYSICS	HEALTH	10	
3	JOB1362684407690	COMP38	MANAGER	DOCTORAL	CHEMISTRY	AUTO	8	
4	JOB1362684407691	COMP7	VICE_PRESIDENT	BACHELORS	PHYSICS	FINANCE	8	

Visualize salary information

In [146]:
```
plt.figure(figsize = (14, 6))
plt.subplot(1,2,1)
sns.boxplot(train_df.salary)
plt.subplot(1,2,2)
sns.distplot(Salary_Predictions.salary, bins=20)
plt.show()
```



Use IQR rule to identify potential outliers

In [147]:
```
stat = Salary_Predictions.salary.describe()
print(stat)
IQR = stat['75%'] - stat['25%']
upper = stat['75%'] + 1.5 * IQR
lower = stat['25%'] - 1.5 * IQR
print('The upper and lower bounds for suspected outliers are {} and {}.'.format(upper, lowe
r))
```

```
count    1000000.000000
mean         116.061818
std           38.717936
min            0.000000
25%           88.000000
50%          114.000000
75%          141.000000
max          301.000000
Name: salary, dtype: float64
The upper and lower bounds for suspected outliers are 220.5 and 8.5.
```

Examine potential outliers

In [148]:
```
#outliers below lower bound
Salary_Predictions[Salary_Predictions.salary < 8.5]
```

Out[148]:

	jobId	companyId	jobType	degree	major	industry	yearsExperi
30559	JOB1362684438246	COMP44	JUNIOR	DOCTORAL	MATH	AUTO	
495984	JOB1362684903671	COMP34	JUNIOR	NONE	NONE	OIL	
652076	JOB1362685059763	COMP25	CTO	HIGH_SCHOOL	NONE	AUTO	
816129	JOB1362685223816	COMP42	MANAGER	DOCTORAL	ENGINEERING	FINANCE	
828156	JOB1362685235843	COMP40	VICE_PRESIDENT	MASTERS	ENGINEERING	WEB	

In [149]:
```
#outliers above upper bound
Salary_Predictions.loc[Salary_Predictions.salary > 222.5, 'jobType'].value_counts()
```

Out[149]:
```
CEO               2893
CFO               1308
CTO               1298
VICE_PRESIDENT     520
MANAGER            188
SENIOR              50
JUNIOR              16
Name: jobType, dtype: int64
```

In [150]:
```
# Check most suspicious potential outliers above upper bound
Salary_Predictions[(Salary_Predictions.salary > 222.5) & (Salary_Predictions.jobType == 'JU
NIOR')]
```

Out[150]:

	jobId	companyId	jobType	degree	major	industry	yearsExperience	milesI
1222	JOB1362684408909	COMP40	JUNIOR	MASTERS	COMPSCI	OIL	24	
27710	JOB1362684435397	COMP21	JUNIOR	DOCTORAL	ENGINEERING	OIL	24	
31355	JOB1362684439042	COMP45	JUNIOR	DOCTORAL	COMPSCI	FINANCE	24	
100042	JOB1362684507729	COMP17	JUNIOR	DOCTORAL	BUSINESS	FINANCE	23	
160333	JOB1362684568020	COMP18	JUNIOR	DOCTORAL	BUSINESS	FINANCE	22	
303778	JOB1362684711465	COMP51	JUNIOR	MASTERS	ENGINEERING	WEB	24	
348354	JOB1362684756041	COMP56	JUNIOR	DOCTORAL	ENGINEERING	OIL	23	
500739	JOB1362684908426	COMP40	JUNIOR	DOCTORAL	ENGINEERING	OIL	21	
627534	JOB1362685035221	COMP5	JUNIOR	DOCTORAL	ENGINEERING	OIL	24	
645555	JOB1362685053242	COMP36	JUNIOR	DOCTORAL	BUSINESS	FINANCE	24	
685775	JOB1362685093462	COMP38	JUNIOR	BACHELORS	ENGINEERING	OIL	24	
743326	JOB1362685151013	COMP14	JUNIOR	DOCTORAL	BUSINESS	FINANCE	19	
787674	JOB1362685195361	COMP43	JUNIOR	DOCTORAL	BUSINESS	FINANCE	18	
796956	JOB1362685204643	COMP30	JUNIOR	MASTERS	BUSINESS	OIL	24	
855219	JOB1362685262906	COMP13	JUNIOR	MASTERS	ENGINEERING	OIL	22	
954368	JOB1362685362055	COMP11	JUNIOR	DOCTORAL	BUSINESS	OIL	24	

The entries with zero salaries do not appear to be volunteer positions. We are confident that they are instances of missing/corrupt data and should be removed from the training set. The high-salary potential outliers appear to be legitimate. Most roles are C-level executive roles and the junior positions are in industries that are well known for high salaries (oil, finance). We determine these entries to be legitimate and will not remove them.

In [151]:
```
# Remove data with zero salaries
Salary_Predictions = Salary_Predictions[Salary_Predictions.salary > 8.5]
```

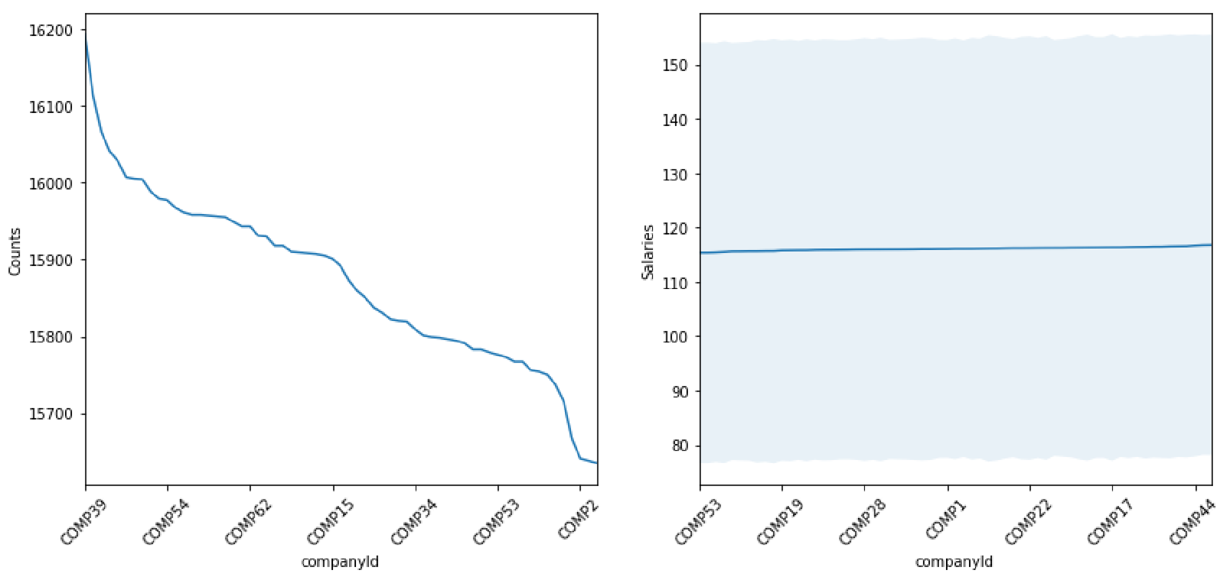
Exploratory Data Analysis

In [152]:

```
def plot_feature(df, col):
    '''
    Make plot for each features
    left, the distribution of samples on the feature
    right, the dependance of salary on the feature
    '''
    plt.figure(figsize = (14, 6))
    plt.subplot(1, 2, 1)
    if df[col].dtype == 'int64':
        df[col].value_counts().sort_index().plot()
    else:
        #change the categorical variable to category type and order their level by the mean
salary
        #in each category
        mean = df.groupby(col)['salary'].mean()
        df[col] = df[col].astype('category')
        levels = mean.sort_values().index.tolist()
        df[col].cat.reorder_categories(levels, inplace=True)
        df[col].value_counts().plot()
    plt.xticks(rotation=45)
    plt.xlabel(col)
    plt.ylabel('Counts')
    plt.subplot(1, 2, 2)

    if df[col].dtype == 'int64' or col == 'companyId':
        #plot the mean salary for each category and fill between the (mean - std, mean + st
d)

        mean = df.groupby(col)['salary'].mean()
        std = df.groupby(col)['salary'].std()
        mean.plot()
        plt.fill_between(range(len(std.index)), mean.values-std.values, mean.values + std.v
alues, \
                         alpha = 0.1)
    else:
        sns.boxplot(x = col, y = 'salary', data=df)
    plt.xticks(rotation=45)
    plt.ylabel('Salaries')
    plt.show()
```

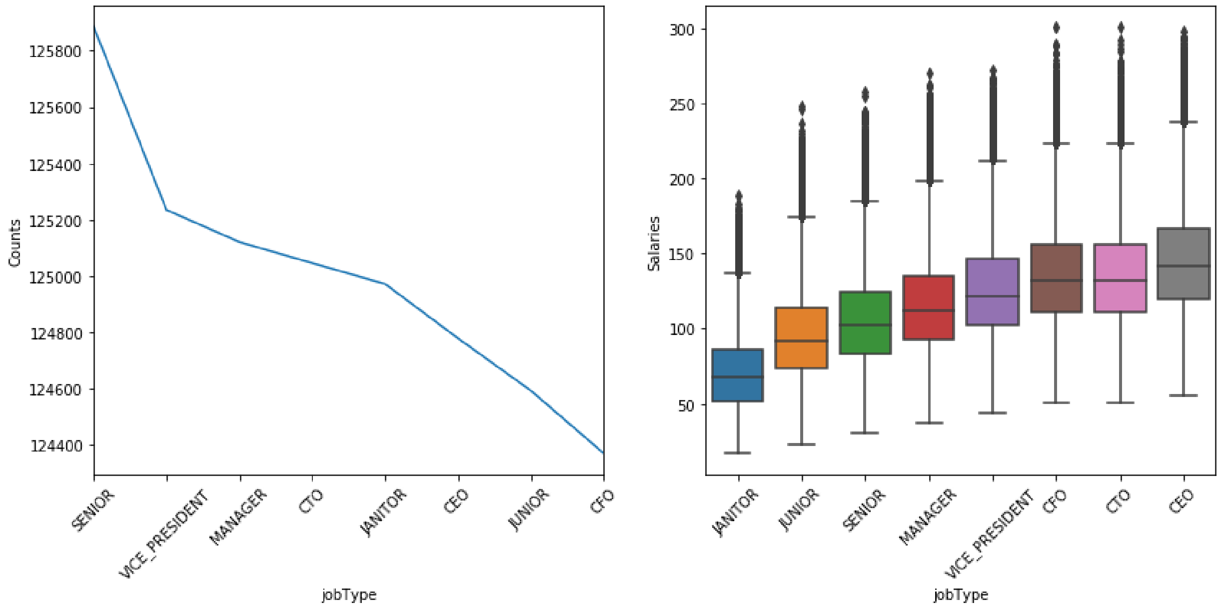
In [153]:

```
plot_feature(Salary_Predictions, 'companyId')
```



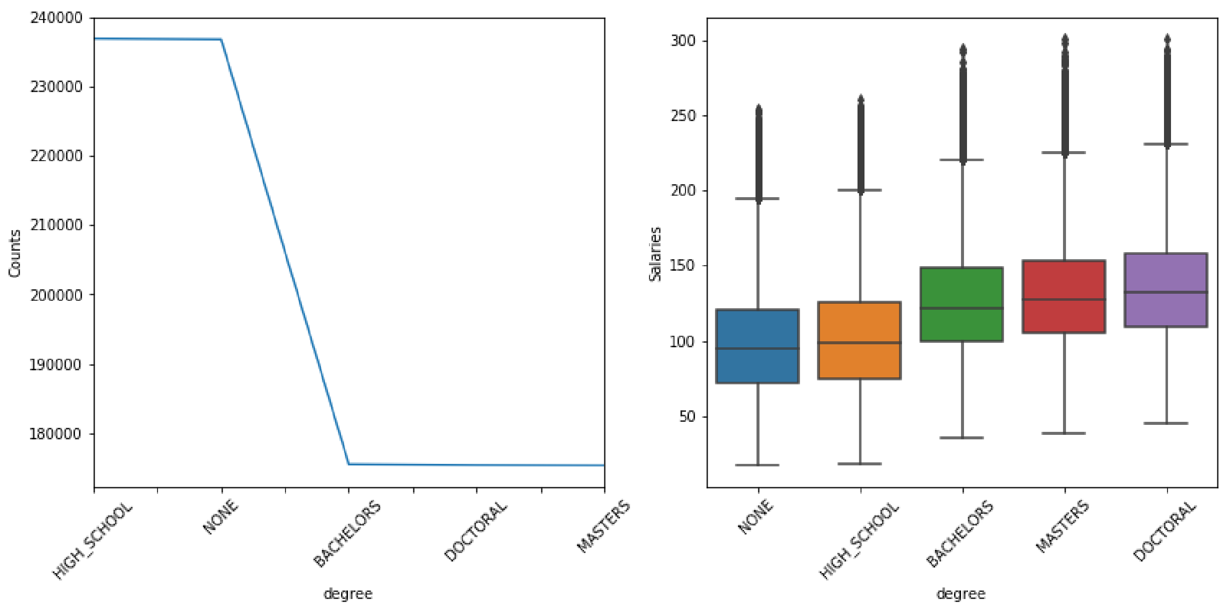
The salary is weakly associated with companies.

In [42]:
```
plot_feature(Salary_Predictions, 'jobType')
```



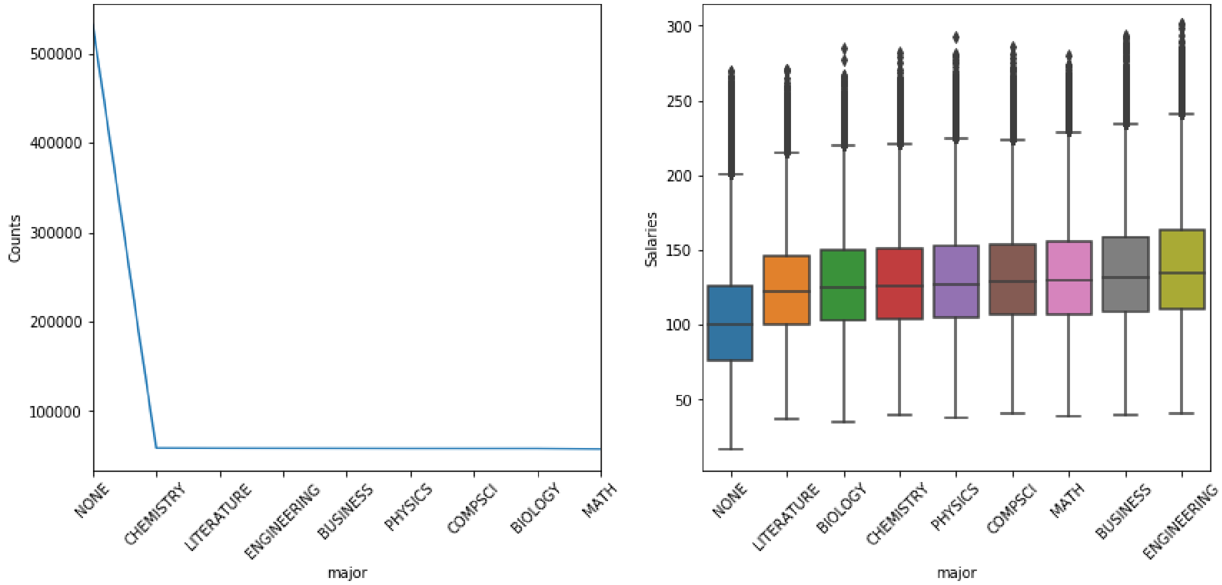
There is a clear positive correlation between job type and salary.

In [43]:
```
plot_feature(Salary_Predictions, 'degree')
```



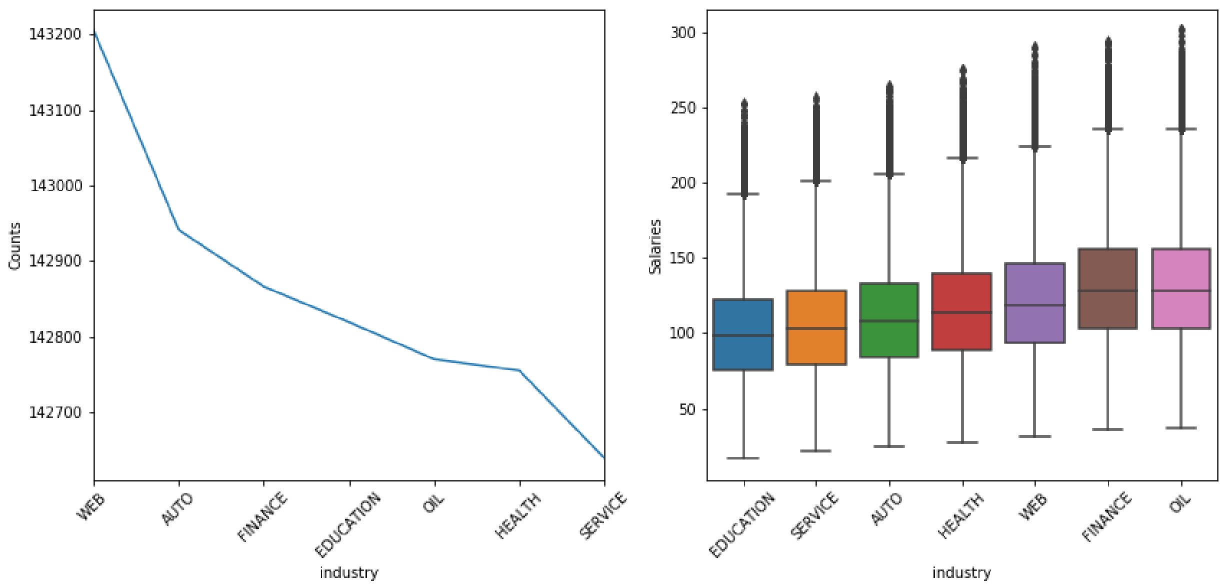
More advanced degrees tend to correspond to higher salaries.

```
In [44]: plot_feature(Salary_Predictions, 'major')
```



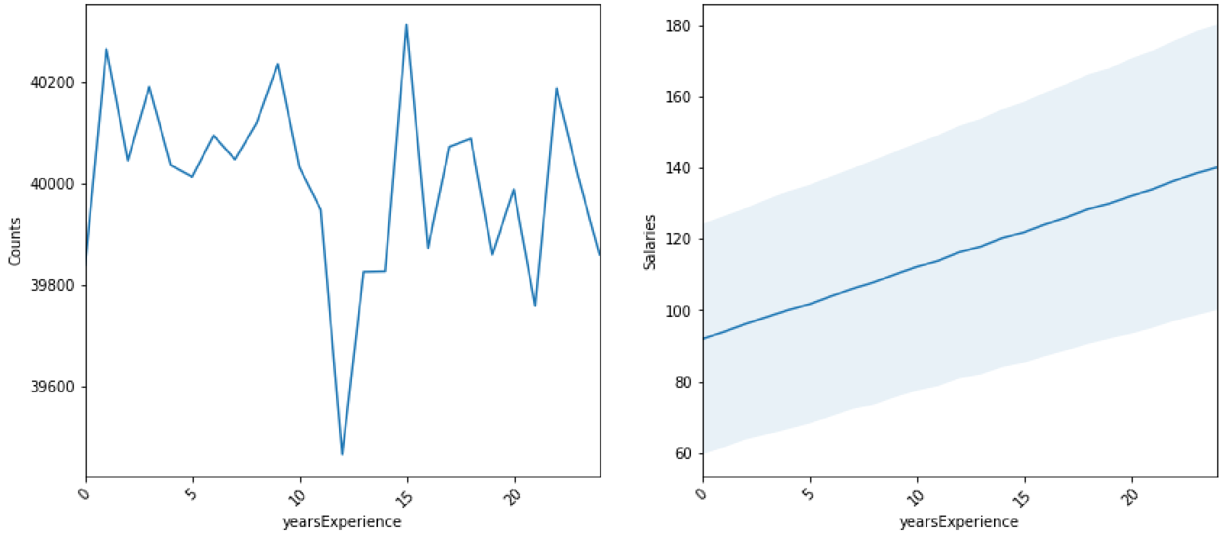
People with majors of engineering, business and math generally have higher salaries.

```
In [45]: plot_feature(Salary_Predictions, 'industry')
```



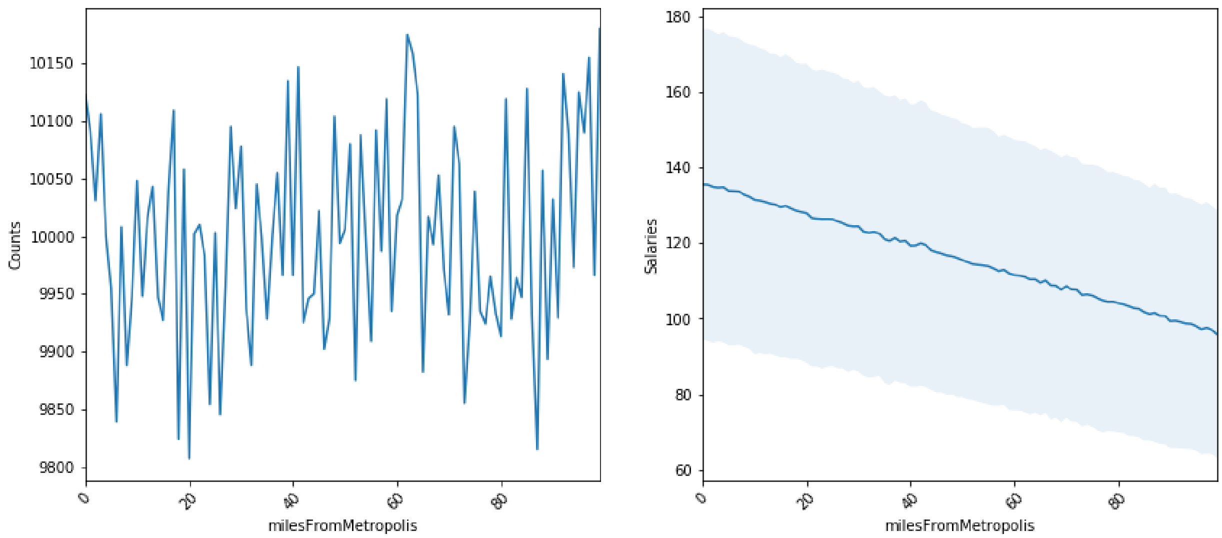
Oil, Finance and Web industries generally have higher salaries.

In [46]:
```python
plot_feature(Salary_Predictions, 'yearsExperience')
```



There is a clear correlation between salary and years of experience.

In [47]:
```python
plot_feature(Salary_Predictions, 'milesFromMetropolis')
```



Salaries decrease with the distance to metropolis.

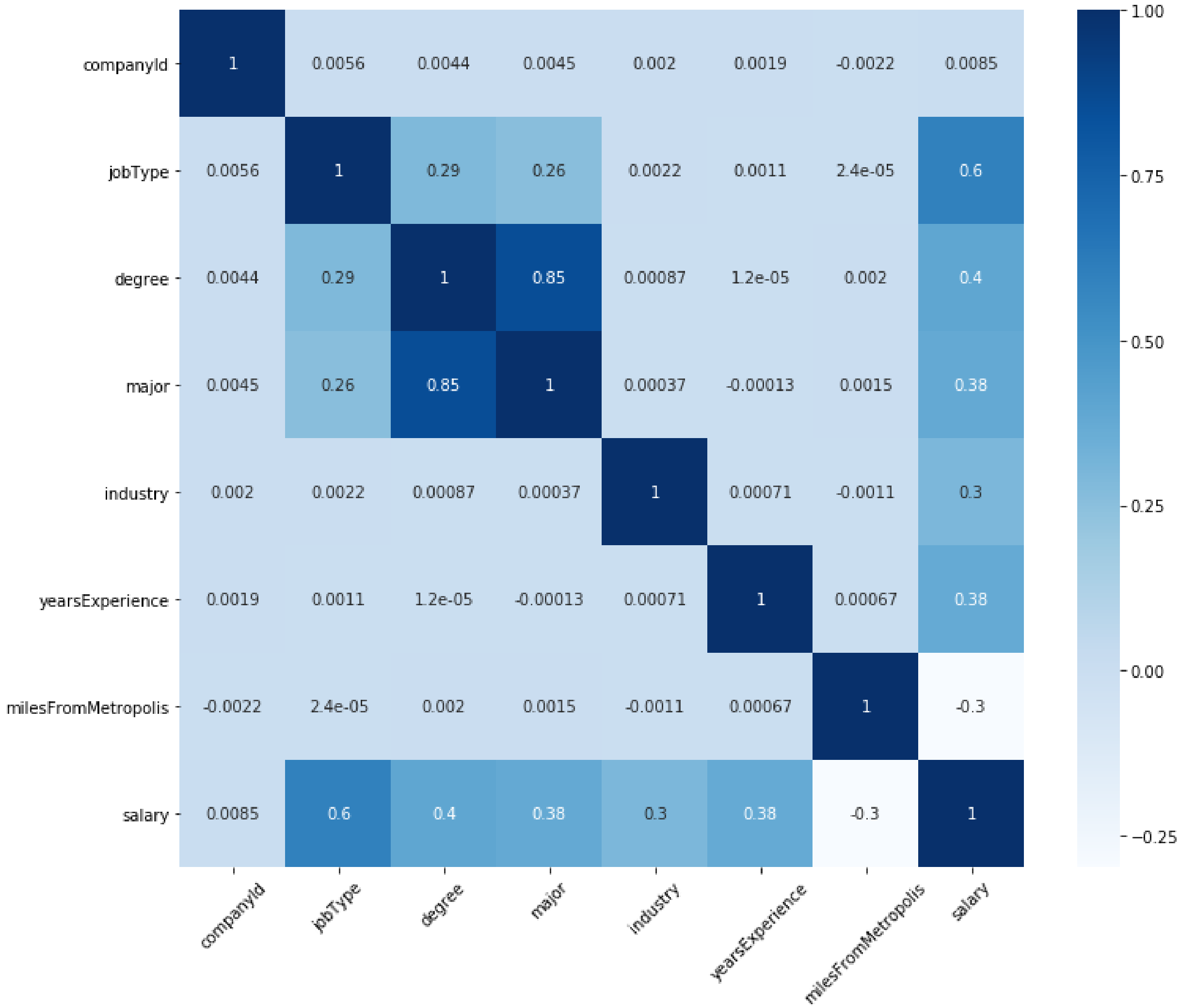
In [58]:
```python
def encode_label(df, col):

    cat_dict ={}
    cats = df[col].cat.categories.tolist()
    for cat in cats:
        cat_dict[cat] = Salary_Predictions[Salary_Predictions[col] == cat]['salary'].mean()
    df[col] = df[col].map(cat_dict)
```

In [63]:
```python
for col in Salary_Predictions.columns:
    if Salary_Predictions[col].dtype.name == "category":
        encode_label(Salary_Predictions, col)
```

Correlations between selected features and response

JobID is discarded because it is unique for individual

In [64]:
```
fig = plt.figure(figsize=(12, 10))
features = ['companyId', 'jobType', 'degree', 'major', 'industry', 'yearsExperience', 'mile
sFromMetropolis']
sns.heatmap(Salary_Predictions[features + ['salary']].corr(), cmap='Blues', annot=True)
plt.xticks(rotation=45)
plt.show()
```



We see that jobType is most strongly correlated with salary, followed by degree, major, and yearsExperience. Among the features, we see that degree and major have a strong degree of correlation and jobType has a moderate degree of correlation with both degree and major.