VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COURSE: PROBABILITY AND STATISTIC
Code: MT2013

SOFTWARE DEVELOPERS SALARY ANALYSIS

ASSIGNMENT REPORT

Group: CC03 - 12

Instructor - Supervisor: PhD. Phan Thi Huong

Ho Chi Minh City, April 26th, 2023

Table of contribution:

	Full name	Student ID	Contribution
1	Trần Duy Minh (Leader) - email: minh.tranduy2209@hcmut.edu.vn	2152773	20%
2	Lê Quang Việt	2153095	20%
3	Phan Lê Tiến Thuận	2153013	20%
4	Đỗ Lâm Ngọc Thức	2153143	20%
5	Nguyễn Minh Triết	2153915	20%

Comment and Evaluation:

Evaluation	Comments

Examiner
(Signature and Full name)

Contents

Acknowledgement

We would like to express our gratitude towards our instructor and supervisor: PhD. Phan Thi Huong for helping us to coordinate in completing this project. We have explored meaningful findings during this work. We are also very grateful to article writers on the internet for providing priceless knowledge and advice about the field we are doing. Without the presence their contributions, this project report would not be done completely.

Disclaimer

For a concise explanation, only the snippets containing necessary calculations are shown and may differ from the final code. Comments are minimized for readability. Due to page limitations, the report only covers some of the source codes' contents. Please only run the codes provided in the provided link for examination. Readers may look up essential installation, required libraries, and guidelines in the same section.

1. Introduction

Statistics and probability are the branches of mathematics that deal with the collection and analysis of data. Probability is the study of chance, a fundamental discipline we use daily. While statistics are more concerned with how data is processed using various methods of analysis and collection techniques. In this project report, the author group will give solutions for problems involving: Descriptive statistics and Inferential statistics. This report also provides code for building different prediction and classification models using statistical optimization methods and machine learning. Predictive modeling is a mathematical process used to predict future events or outcomes by analyzing patterns in a given input data set. It is a crucial part of predictive analysis, which uses current and historical data to forecast activity, behavior, and trends in business and research.

1.1. Data description

The table below contains basic information of the dataset:

Name:	U.S. Software Developer Salaries
Source:	U.S. Software Developer Salaries | Kaggle
Purpose:	This dataset provides an extensive look into the financial health of software developers in major cities and metropolitan areas around the United States. We explore disparities between states and cities in terms of mean software developer salaries, median home prices, cost of living averages, rent averages, cost of living plus rent averages and local purchasing power averages. Through this data set we can gain insights on how to better understand which areas are more financially viable than others when seeking employment within the software development field. Our data allow us to uncover patterns among certain geographic locations to identify other compelling financial opportunities that software developers may benefit from.

1.2. Attribute information

The table below contains attribute information from the dataset:

NAME	**TYPE OF VARIABLE**	**DESCRIPTION**
METRO	CATEGORICAL	The metropolitan area of the city
adjusted_dev_salary	CONTINUOUS	The average salary for software developers adjusted for cost-of-living differences between cities
unadjusted_dev_salary	CONTINUOUS	The average salary for software developers without adjusting for cost-of-living differences between cities
Mean Unadjusted Salary (all occupations)	CONTINUOUS	The average salary for all occupations without adjusting for cost-of-living differences between cities
num_dev_jobs	CONTINUOUS	The number of software developer jobs in the city
avg_home_price	CONTINUOUS	The median home price in the city
Cost of Living avg	CONTINUOUS	The average cost of living in the city
Rent avg	CONTINUOUS	The average rent in

		the city
living_expenses	CONTINUOUS	The average cost of living plus rent in the city
avg_purchasing_power	CONTINUOUS	The average local purchasing power in the city
state	CATEGORICAL	The state to which the city belongs
region	CATEGORICAL	The region in which the city is located

2. Background

2.1. Purpose of the project

The author group's objective in completing this project is to comprehend the test techniques learned in probability and statistics and be able to use those techniques to analyze data in related fields, including:

- Comparing salaries of software developers in different cities to determine which city provides the best compensation package.
- Estimating the cost of relocating to a new city by looking at average costs such as rent and cost of living.
- Predicting job growth for software developers by analyzing factors like local purchasing power, median home price and number of jobs available.

2.2. Theoretical methodology

2.2.1. Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) is a methodology for obtaining insights from data, often using data visualization and statistical graphics to help identify patterns and trends, detect outliers and understand the relationships between variables. EDA is used to extract essential features for the prediction models. Graphing the raw data allows us to have a general idea of the behavior and distribution of the variables: Histograms, Box plots, Pair plots,…

2.2.2. Analysis of Variance (ANOVA)

An ANOVA test is a statistical test used to determine whether there is a statistically significant difference between two or more categorical groups by testing the difference in means using the variance. In this project, the author groups perform one-way ANOVA tests.

One-way ANOVA has a categorical independent variable (a factor) and a normally distributed continuous dependent variable (such as the range or ratio level). Independent variables divide items into two or more mutually exclusive levels, categories, or groups. Hypothesis testing for ANOVA can generally be stated as follows:

- H0 (Null hypothesis): There is no difference in means between groups of independent subjects.

- H1 (Alternative hypothesis): There are two or more independent groups with statistically different means.

Some assumptions to consider before running an ANOVA test:

1. The sample population is normally distributed.

2. Homogeneity of variance between groups of independent variables (the variance is approximately the same for each group of samples).

3. All sample observations must be independent of each other.

The ANOVA formula:

$$F = \frac{MSB}{MSE} \quad (1)$$

Where:

- ***F*** is ANOVA coefficient
- ***MSB*** is the Mean sum of square due to treatment
- ***MSE*** is the Mean sum of square due to error

The MSB formula:

$$\text{MSB} = \frac{\sum_{i=1}^{k} \quad n_i(\underline{X_i} - \underline{X})^2}{k-1} \quad (2)$$

Where Xi is the mean for each group, and X is the overall mean.

The MSE formula:

$$\text{MSE} = \frac{\sum \quad (X_{ij} - \underline{X_j})^2}{n_{total} - k} \qquad (3)$$

Where $n_{total} = n_1 + n_2 + \ldots + n_i$ with n_i being the sample size of group i

After you have run an ANOVA and found significant results, you can run Tukey's HSD to find out which specific groups' means (compared with each other) are different. Tukey's Honest Significant Difference (HSD) test is a post-hoc test based on the studentized range distribution. The test compares all possible pairs of means. Any absolute difference between means has to exceed the value of HSD to be statistically significant.

The Tukey HSD test statistic formula is:

$$HSD = \frac{M_i - M_j}{\sqrt{\frac{MS_w}{n_h}}}$$

Where:

- ***M_i - M_j*** is the difference between means of pair groups

- MS_w is the Mean Square Within and n is the number in each category

2.2.3. Multivariate Linear Regression (MLR)

Most data analyzed statistically does not necessarily have a response variable or an explanatory variable. Multivariate regression is a technique used to measure the relationship between a continuous dependent variable and two or more independent variables. Relationships that result from correlations between variables are called linear. After applying multivariate regression to a data set, we use this technique to predict the behavior of a response variable based on its predictors.

The null hypothesis and alternative hypothesis could be expressed as:

- H0 (Null hypothesis): There is no relationship between explanatory variables and response variable.

- H1 (Alternative hypothesis): There is at least one explanatory variable is associated linearly with the response variable.

The general equation of MLR:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

Where:

- Y is the dependent variable.
- X_i is the i^{th} independent variable.
- β_0 is the intercept of Y when all X_i are 0s.
- β_i is the coefficient for each X_i.
- ε is called the model's random error (residual) term.

Model performance metrics:

- R-squared (R2): is the portion of the variation in the outcome explained by the predictor, is the squared correlation between the observed outcome value and the value predicted by the model. The higher the value, the better the model.

- Root mean square error (RMSE): measures the average error made by a model in predicting an observation. The lower the RMSE, the better the model.

Diagnostic plots of a linear regression model:

1. Residuals vs. Leverage Plot: This plot is used to identify influential observations. If any point on this graph is outside of Cook's distance (dotted line), it is an influential observation.

2. Scale-Location Plot: This plot is used to test the assumption of equal variance among the residuals of a regression model (homoscedasticity). If the red line is roughly horizontal across the graph, the assumption of equal variances may be correct.

3. Normal Q-Q Plot: This plot is used to determine whether the residuals of a regression model follow a normal distribution. If the points on this graph fall roughly along a straight diagonal line, we can assume that the residuals are normally distributed.

4. Residuals vs. Fitted Plot: This plot is used to determine if the residuals show a non-linear pattern. If the red line in the middle of the graph is roughly horizontal, you can assume that the residuals follow a linear pattern.

3. Descriptive Statistics

3.1. Import Data and Data cleaning

To start working with R studio, we need to import the libraries for the project.

The next step is to load the dataset to data variable and start cleaning the data by removing the unnecessary features and set them to NULL.

The data cleaning step contains two operations:

The first operation is to remove any rows containing missing values (NAs) from the data frame. This is done using the **na.omit()** function, which removes any rows containing NAs and returns the modified data frame. The modified data frame is then assigned back to the same variable `data’.

The second operation is to check whether there are any remaining missing values in the data frame. This is done using the **is.na()** function, which returns a logical vector indicating whether each element of the data frame is missing or not. The **sum()** function is then used to count the number of missing values, by summing up the logical vector obtained from **is.na()**. If there are no missing values in the data frame, the result of **sum(is.na(data))** will be 0.

3.2. Data summary

Firstly, we use the **head**() function, which is an R command that displays the first few rows of the data data frame in the console. By default, the **head()** function shows the first 6 rows of the data frame, along with the column names and the first few values of each column.

This command is often used as a quick way to check the structure and content of a data frame after importing or manipulating data. It can help us to identify any issues with the data, such as missing values or unexpected values in certain columns.

```
> head(data)
  adjusted_dev_salary num_dev_jobs avg_home_price living_expenses
1              117552        13430         192000          2856.5
2              117323        65760         491600          4091.5
3              114122        12800         208500          3221.1
4              112118         5780         296500          3094.5
5              111616         4240         124100          2586.0
6              111050         1560         136000          2888.0
  avg_purchasing_power
1               9335.4
2               8971.3
3               8939.8
4               8493.1
5               4887.7
6               5721.9
```

Then, we use the **summary()** function, which provides a summary of each variable in the `data’ data frame. From the output, we can see that the `data’ data frame has 5 variables: `adjusted_dev_salary’, `num_dev_jobs’, `avg_home_price’, `living_expenses’, and `avg_purchasing_power'.

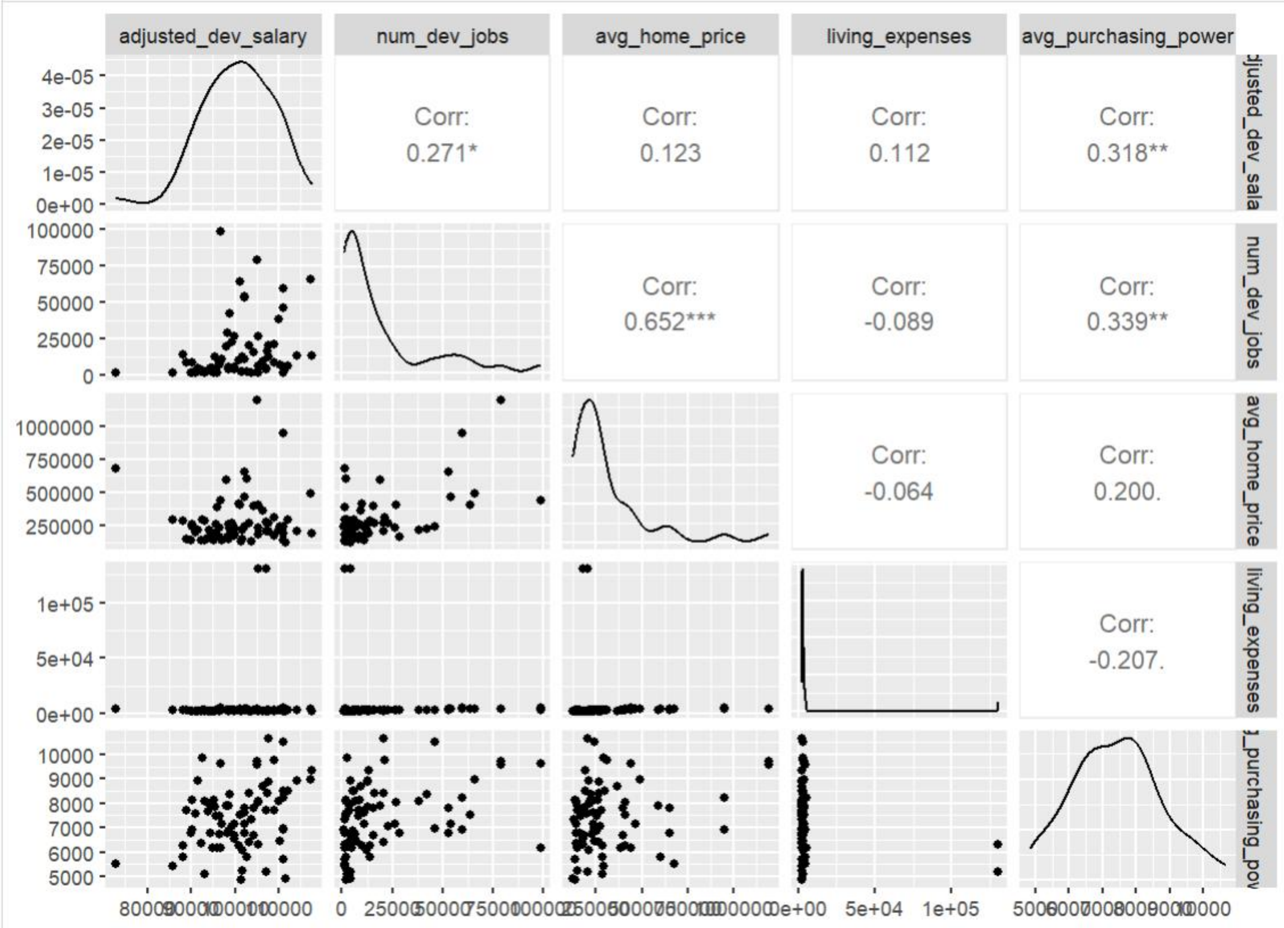
```
> summary(data)
 adjusted_dev_salary  num_dev_jobs    avg_home_price    living_expenses  avg_purchasing_power
 Min.   : 72811      Min.   : 1120   Min.   : 124100   Min.   :   2241   Min.   : 4840
 1st Qu.: 95308      1st Qu.: 3170   1st Qu.: 178400   1st Qu.:   2811   1st Qu.: 6464
 Median :101256      Median : 8770   Median : 243700   Median :   3000   Median : 7499
 Mean   :100833      Mean   :19099   Mean   : 312701   Mean   :   6507   Mean   : 7421
 3rd Qu.:107170      3rd Qu.:21160   3rd Qu.: 366000   3rd Qu.:   3447   3rd Qu.: 8153
 Max.   :117552      Max.   :98650   Max.   :1193600   Max.   :130098   Max.   :10674
```

For the numeric variables, we can see the minimum, maximum, median, mean, and quartile values. For example, we can see that the `adjusted_dev_salary' variable has a minimum value of 72811, a maximum value of 117552, a median value of 101256, a mean value of 100833, and quartiles at 95308, 101256, and 107170. Similarly, the **summary()** function also gives an overview of the remaining variables.

These summary statistics are useful for getting a quick understanding of the distribution of values in each variable in the data data frame. For instance, we can see that the mean value of `living_expenses' is 6507, which is higher than the median value of 3000, suggesting that the distribution of living expenses is skewed.

3.3. Data visualization

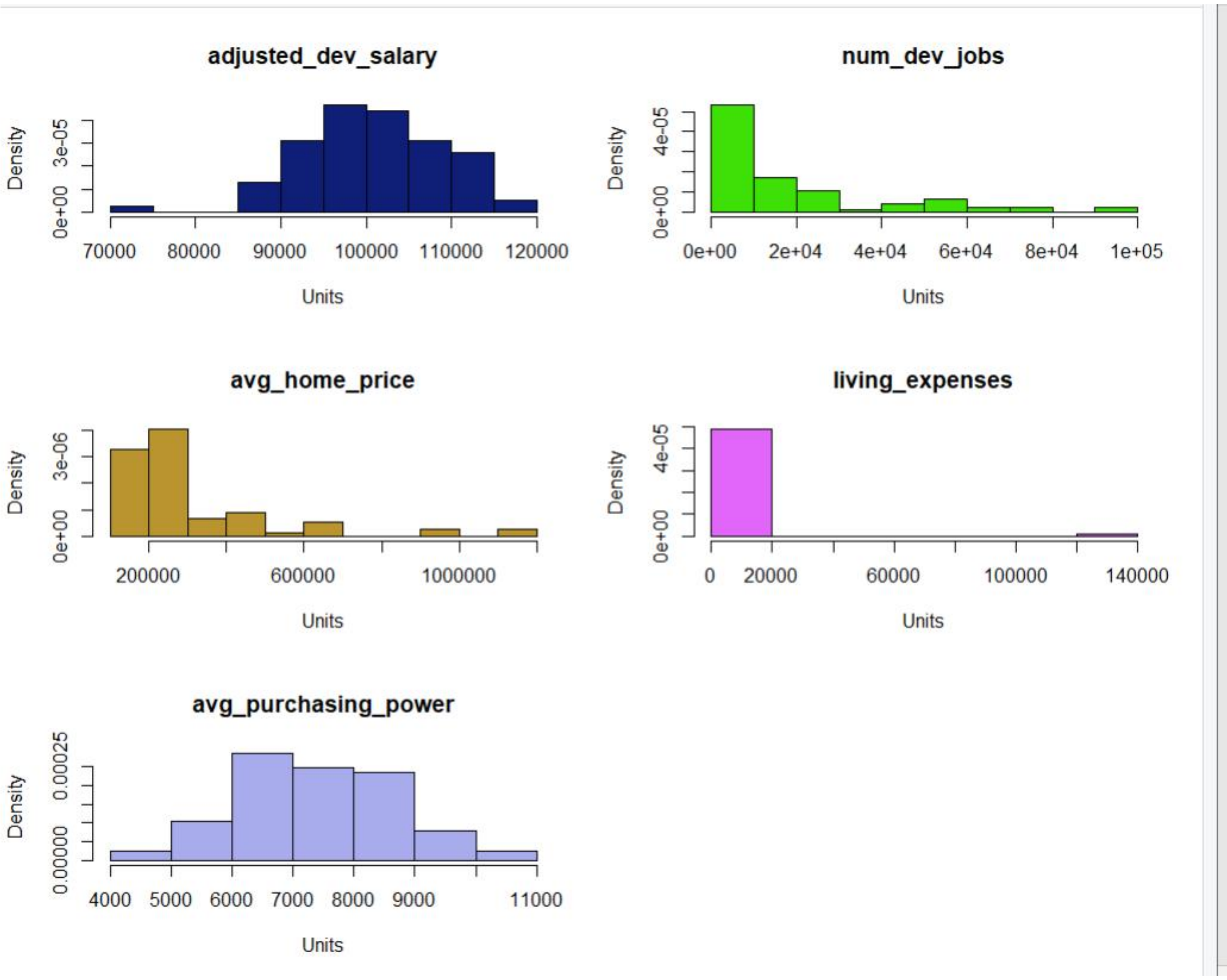
We use the **ggpairs()** function to create a plot matrix that allows us to visualize the relationships and distributions of all pairs of variables in the `data' data frame. This can be a useful tool for identifying patterns and relationships in the data and exploring potential correlations between variables.



The correlation values are near to +-1, it indicates a strong linear relationship between the variables, which means that the `Corr' values closer to 1 indicates that the two variables tend to increase or decrease together, while a value closer to -1 indicates that the two variables tend to have an inverse relationship, where one increases as the other decreases . On the other hand, low correlations between variables may suggest that they are unrelated, and may not be useful predictors in a model.

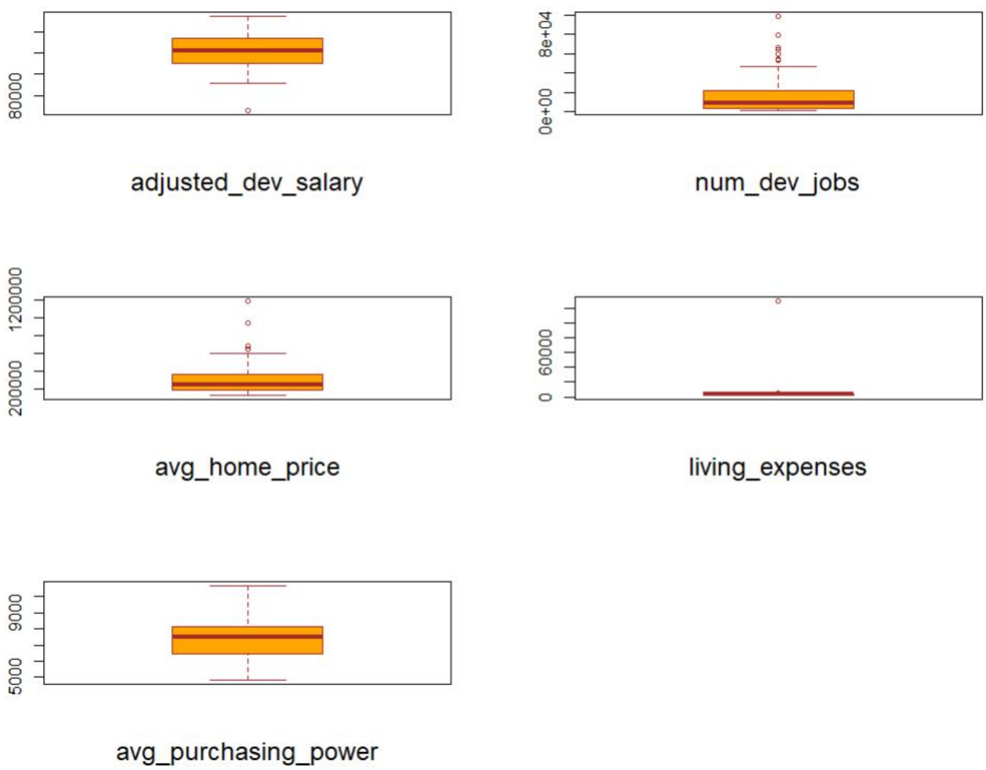
From the plot matrix of all pairs of variables, the correlation coefficient between variables `avg_home_price' and `num_dev_jobs' is 0.652, indicating a moderate possitive correlation between the two variables. Meanwhile, there seems to not be a relationship between some pairs, such as: `living_expenses' and `num_dev_jobs' (the `Corr' value is -0.089), or `living_expenses' and `avg_home_price'(the `Corr' value is -0.064).

We use the **hist()** function to create a histogram, which is a graphical representation of the distribution of a numeric variable. It takes several arguments, such as the data to be plotted, the main title, the color, and the x-axis label. The `freq’ argument is set to `FALSE’ to plot the histogram as a probability density function instead of a frequency distribution. It is a useful tool for visualizing the distribution of a variable, including its central tendency, spread, and skewness.



It is evident from the graph that the distributions of `num_dev_jobs’, `avg_home_price’, and `living_expenses’ are positive-skewed, indicating a concentration of values towards the lower end of the range, with some extreme values to the right. On the other hand, the distributions of `adjusted_dev_salary’ and `avg_purchasing_power’ are approximately symmetric, suggesting that values are more evenly distributed around the central tendency.

We decide to use a boxplot, which provides a visual representation of the distribution of a dataset through its quartiles, the minimum and maximum values, and any outliers. To do that, we use the **boxplot()** function to create a boxplot for each variable in the dataset.



It is obvious that most variables have outliers, particularly `num_dev_jobs’ and `avg_home_price’, which have 3 and 6 outliers, respectively. The only variable that does not have any apparent outliers is `avg_purchasing_power’.

Moreover, the boxplot also indicates the high concentration of the variable `living_expenses' through its median and interquartile range, and similarly for other variables in their respective graphs.

4. Inferential Statistics:

4.1. Comparing salaries of software developers in different cities

In this code, we add a variable `highest_salary_city’ and use the **which.max()** function to find the index of the highest value in the `adjusted_dev_salary’ column of the `data’ dataset. It then uses this index to select the corresponding city from the city column and assigns it to the highest_salary_city variable.

```
The city with the highest compensation is Columbus, OH  with the
salary of 117552
```

Then, we use the **cat()** function to print a statement that includes the name of the city with the highest compensation and its corresponding salary. The `highest_salary_city' variable and the highest salary value from the `adjusted_dev_salary' column are used to complete the statement. The `\n' character is included at the end of the statement to start a new line.

```
The city with the lowest compensation is Honolulu, HI  with the
salary of 72811
```

This code is very similar to the previous code, except it uses the **which.min()** function instead of **which.max()** to find the index of the lowest value in the `adjusted_dev_salary' column of the data dataset.

Based on the output, we can conclude that there is a significant difference in compensation between the cities included in the analysis. `Columbus, OH' has the highest compensation with a salary of 117552, while `Honolulu, HI' has the lowest compensation with a salary of 72811. It would be interesting to investigate further why there is such a difference in compensation between these cities and whether there are any factors contributing to it.

4.2. Estimating the cost of relocating to a new city

The program includes several steps for estimating the total cost of relocating for software. First, the dataset is read in and the average cost of living across the US is calculated. Then the dataset is filtered to include only the featured states, in the example are California, New York, and Texas; and the average salary is calculated for each state. The cost of living and housing data are added for each state, and the total estimated cost of relocating is calculated based on the adjusted salary, cost of living, and rent cost. Finally, a bar plot is created to display the total cost of relocating by state.

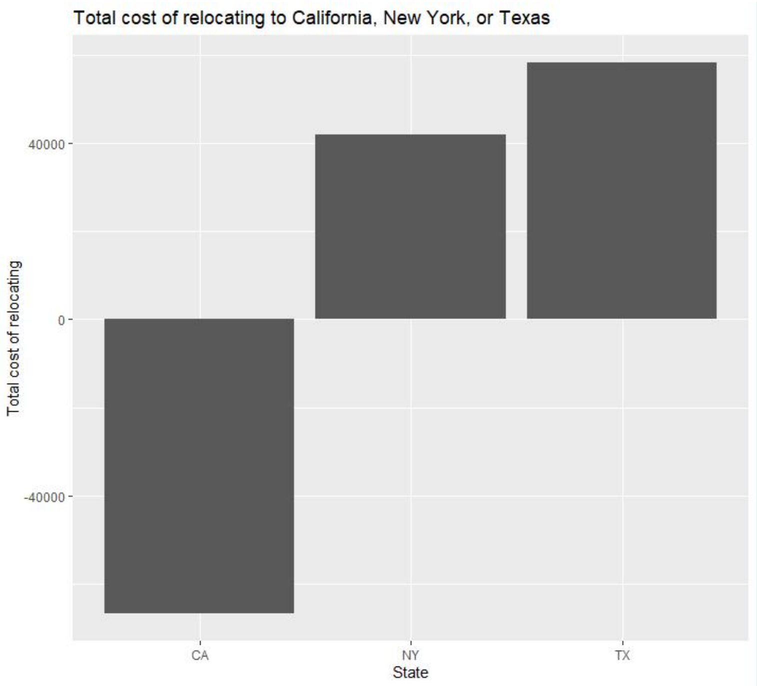
To estimate the total cost of relocating to a new state given salary, cost of living, and rent cost of that state, we can use the following formula:

Total cost of relocating = Salary - (Salary * Cost of living index / 100) + Rent cost * 12

Here, the "Salary" is the annual IT salary of the person who is relocating, the "Cost of living index" determines the ratio between cost of living in that state and the median of the whole country, and the "Rent cost" is the monthly cost of rent.

If the total cost of relocating is negative, indicating that the cost of living and rent expenses are expected to exceed the person's salary.

We take 3 states California, New York and Texas for demonstration



As shown, people relocating to California will experience higher cost of living and cost of rent respected to their annual salary. Those costs may exceed the person's salary. This shows the importance of careful financial planning and research before making a decision to relocate.

Using the mean cost of living for the whole country, we calculate the total estimated cost of relocating for each state, which includes the adjusted salary minus the cost of living adjustment and the rent cost for a year.

The resulting plot shows that relocating to California would be the most expensive, followed by New York and then Texas. This suggests that developers may need to factor in higher costs of living and rent when considering relocating to California or New York compared to Texas.

To help software developers who want to relocate to a new city or state, we can compare the total cost of living in the featured states and other popular tech hubs, such as Seattle, Boston, or San Francisco. This way, they can see how much income they need to maintain their standard of living in different places. For example, according to NerdWallet, a person who earns $100,000 in San Francisco (CA) would need $81,654 in Seattle (WA), $81,389 in Boston (MA) to have the same quality of life. By providing this information, we can offer more comprehensive and useful guidance to developers.

4.3. Predicting job growth for software developers

4.3.1. Salary and Number of Developer Jobs

In terms of Salary and Number of jobs, we want to determine if the difference in occupation available affects the salary. To use the number of developer jobs as a factor, we divided it into segments.

Hypothesis:

• H0: The mean salary is the same for all numbers of jobs.

• H1: At least one segment for the figure of occupation available has a mean salary that is different from the other.

We use aov() to store the test result into a one-way variable, then use summary() to summarize the information:

```
                         Df    Sum Sq   Mean Sq F value   Pr(>F)    
as.factor(num_dev_jobs) 10 1.054e+10 1.054e+09   16.55 1.87e-14 ***
Residuals               66 4.203e+09 6.367e+07                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The condition variable's p-value is low (indicated by the '***'), implying that the difference in available jobs influences the mean salary. As a result, H0 is rejected. The average salary of software developers varies significantly depending on the provided number of developer jobs in each metropolitan area.

Tukey’HSD test to determine the difference between each two segments of the figure for jobs available:

	diff	lwr	upr	p adj
> 90000-< 5000	25968.643	6495.94444	45441.341	0.0016059
10000-15000-< 5000	14767.643	-12308.04554	41843.331	0.7649926
10000-20000-< 5000	9821.976	642.45812	19001.494	0.0261669
20000-30000-< 5000	11879.893	1214.25668	22545.529	0.0170498
30000-40000-< 5000	14055.643	-13020.04554	41131.331	0.8141758
40000-50000-< 5000	15824.310	-337.89983	31986.519	0.0601094
5000-10000-< 5000	4931.797	-3997.16089	13860.754	0.7512217
50000-60000-< 5000	35272.243	22355.51599	48188.970	0.0000000
60000-70000-< 5000	33208.143	13735.44444	52680.841	0.0000167
70000-80000-< 5000	45118.643	25645.94444	64591.341	0.0000000
10000-15000-> 90000	-11201.000	-43785.05676	21383.057	0.9859361
10000-20000-> 90000	-16146.667	-36466.39611	4173.063	0.2457501
20000-30000-> 90000	-14088.750	-35121.66819	6944.168	0.4905242
30000-40000-> 90000	-11913.000	-44497.05676	20671.057	0.9780149
40000-50000-> 90000	-10144.333	-34431.05530	14142.389	0.9459810
5000-10000-> 90000	-21036.846	-41244.62031	-829.072	0.0343995
50000-60000-> 90000	9303.600	-12955.54836	31562.748	0.9457522
60000-70000-> 90000	7239.500	-19365.27093	33844.271	0.9977631
70000-80000-> 90000	19150.000	-7454.77093	45754.771	0.3826526
10000-20000-10000-15000	-4945.667	-32636.79020	22745.457	0.9999453
20000-30000-10000-15000	-2887.750	-31106.37091	25330.871	0.9999997
30000-40000-10000-15000	-712.000	-38336.82788	36912.828	1.0000000
40000-50000-10000-15000	1056.667	-29663.87665	31777.210	1.0000000
5000-10000-10000-15000	-9835.846	-37444.92174	17773.229	0.9817407
50000-60000-10000-15000	20504.600	-8639.46636	49648.666	0.4165993
60000-70000-10000-15000	18440.500	-14143.55676	51024.557	0.7232704
70000-80000-10000-15000	30351.000	-2233.05676	62935.057	0.0898717
20000-30000-10000-20000	2057.917	-10085.44431	14201.278	0.9999664
30000-40000-10000-20000	4233.667	-23457.45687	31924.790	0.9999872
40000-50000-10000-20000	6002.333	-11170.97246	23175.639	0.9841083
5000-10000-10000-20000	-4890.179	-15540.61162	5760.253	0.9038791
50000-60000-10000-20000	25450.267	11288.79592	39611.737	0.0000050
60000-70000-10000-20000	23386.167	3066.43722	43705.896	0.0117254
70000-80000-10000-20000	35296.667	14976.93722	55616.396	0.0000111
30000-40000-20000-30000	2175.750	-26042.87091	30394.371	1.0000000
40000-50000-20000-30000	3944.417	-14067.09840	21955.932	0.9996536
5000-10000-20000-30000	-6948.096	-18903.17657	5006.984	0.6907362
50000-60000-20000-30000	23392.350	8225.29700	38559.403	0.0001322
60000-70000-20000-30000	21328.250	295.33181	42361.168	0.0440742
70000-80000-20000-30000	33238.750	12205.83181	54271.668	0.0000822
40000-50000-30000-40000	1768.667	-28951.87665	32489.210	1.0000000
5000-10000-30000-40000	-9123.846	-36732.92174	18485.229	0.9895291
50000-60000-30000-40000	21216.600	-7927.46636	50360.666	0.3660206
60000-70000-30000-40000	19152.500	-13431.55676	51736.557	0.6763167
70000-80000-30000-40000	31063.000	-1521.05676	63647.057	0.0749557
5000-10000-40000-50000	-10892.513	-27933.20423	6148.179	0.5601314
50000-60000-40000-50000	19447.933	18.55576	38877.311	0.0495759
60000-70000-40000-50000	17383.833	-6902.88863	41670.555	0.3909401
70000-80000-40000-50000	29294.333	5007.61137	53581.055	0.0065788
50000-60000-5000-10000	30340.446	16340.08953	44340.803	0.0000000
60000-70000-5000-10000	28276.346	8068.57200	48484.120	0.0007482
70000-80000-5000-10000	40186.846	19979.07200	60394.620	0.0000004
60000-70000-50000-60000	-2064.100	-24323.24836	20195.048	0.9999999
70000-80000-50000-60000	9846.400	-12412.74836	32105.548	0.9228828
70000-80000-60000-70000	11910.500	-14694.27093	38515.271	0.9171463

There are statistically significant differences ($p < 0.05$) between several segment groups, for example: over 90000 and below 5000 jobs, 10000-20000 group and below 5000, 20000-3000 group and below 5000, 70000-80000 group and 10000-20000 group, 50000-60000 group and 20000-30000 group, 70000-80000 and 5000-10000 group. The other group differences are not statistically significant.

We continue to use the Levene test to verify the null hypothesis that the population variances are equal

```
> leveneTest(unadjusted_dev_salary ~ as.factor(num_dev_jobs), data = sD)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group 10   1.453 0.1774
      66
```

Calculating the tests yields a p value of 0.1774, which is greater than the 5% level of significance. As a result, there is no reason to reject the null hypothesis H0. We can say that the variance between groups appears to be equal.

To make the result more clear, we will apply the Shapiro-Wilk test to assess the normality assumption based on the residuals.

```
> aov_residuals <- residuals(object = one_way_aov )
> shapiro.test(x = aov_residuals )

        Shapiro-Wilk normality test

data:  aov_residuals
W = 0.98975, p-value = 0.7988
```

We can approve H0 and state that no indication that normality is violated.

4.3.2. Salary and Average Purchasing Power

In terms of Salary and Average Purchasing Power, we want to determine if the difference in the purchase capacity of citizens who have an occupation in the field of software developer affects their salary. To use the figure of average purchasing power as a factor, we divided it into groups which each cover a segment of 1000.

Hypothesis:

• H0: The salary level is the same for all mean purchase power.

• H1: At least one segment for the figure of average purchase capacity has a mean salary that is different from the other.

We use aov() to store the test result into a one-way variable, then use summary() to summarize the information:

```
                                Df    Sum Sq   Mean Sq F value Pr(>F)
as.factor(avg_purchasing_power)  6 2.368e+09 394749956   2.234 0.0498 *
Residuals                       70 1.237e+10 176739885
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The condition variable's p-value is low (indicated by the '***'), implying that the difference in the average purchase power influences the mean salary. As a result, H0 is rejected. The average salary of software developers varies significantly depending on the mean purchasing capacity in each metropolitan area.

Tukey’HSD test to determine the difference between each two segments of the figure for average purchasing power:

```
                            diff         lwr      upr     p adj
4000-5000-> 10000    -11479.0000 -51836.9683 28878.97 0.9767428
5000-6000-> 10000    -14984.3750 -46890.1504 16921.40 0.7861366
6000-7000-> 10000    -10673.0909 -40479.4258 19133.24 0.9299006
7000-8000-> 10000    -11206.0526 -41207.8410 18795.74 0.9153686
8000-9000-> 10000     -5719.3333 -35800.3868 24361.72 0.9972715
9000-10000-> 10000     7247.6667 -25704.4764 40199.81 0.9939388
5000-6000-4000-5000   -3505.3750 -35411.1504 28400.40 0.9998818
6000-7000-4000-5000     805.9091 -29000.4258 30612.24 1.0000000
7000-8000-4000-5000     272.9474 -29728.8410 30274.74 1.0000000
8000-9000-4000-5000    5759.6667 -24321.3868 35840.72 0.9971631
9000-10000-4000-5000  18726.6667 -14225.4764 51678.81 0.6018220
6000-7000-5000-6000    4311.2841 -12350.9637 20973.53 0.9856607
7000-8000-5000-6000    3778.3224 -13231.0928 20787.74 0.9936065
8000-9000-5000-6000    9265.0417  -7883.7973 26413.88 0.6571840
9000-10000-5000-6000  22232.0417    436.2477 44027.84 0.0426236
7000-8000-6000-7000    -532.9617 -13172.5545 12106.63 0.9999996
8000-9000-6000-7000    4953.7576  -7872.8467 17780.36 0.9020917
9000-10000-6000-7000  17920.7576   -666.7580 36508.27 0.0660063
8000-9000-7000-8000    5486.7193  -7787.7480 18761.19 0.8697356
9000-10000-7000-8000  18453.7193   -445.6309 37353.07 0.0600879
9000-10000-8000-9000  12967.0000  -6057.9287 31991.93 0.3822128
```

There are statistically significant differences ($p < 0.05$) between segment groups: 9000-10000 group and 5000-6000 group. The other group differences are not statistically significant.

We continue to use the Levene test to verify the null hypothesis that the population variances are equal

```
> leveneTest(unadjusted_dev_salary ~ as.factor(avg_purchasing_power), data = sD)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  6  0.9778 0.4469
      70
```

Calculating the tests yields a p value of 0.4469, which is greater than the 5% level of significance. As a result, there is no reason to reject the null hypothesis H0. We can say that the variance between groups appears to be equal.

To make the result more clear, we will apply the Shapiro-Wilk test to assess the normality assumption based on the residuals.

```
> aov_residuals_2 <- residuals(object = one_way_aov_2 )
> shapiro.test(x = aov_residuals_2 )

        Shapiro-Wilk normality test

data:  aov_residuals_2
W = 0.94823, p-value = 0.003429
```

The p-value is approximately zero, thus we can reject H0 and state that the data is non-normal at high certainty.

4.3.3. Salary and Average Home Price

In terms of Salary and Average Home Price, we want to determine if the difference in the accommodation expenses affects the salary. To use the figure of average home price as a factor, we divided it into groups of value.

Hypothesis:

• H0: The salary level is the same for all mean accommodation costs.

• H1: At least one segment for the figure of average home price has a mean salary that is different from the other.

We use aov() to store the test result into a one-way variable, then use summary() to summarize the information:

```
                       Df    Sum Sq   Mean Sq F value  Pr(>F)   
as.factor(avg_home_price)  2 1.912e+09 955786583   5.513 0.00586 **
Residuals              74 1.283e+10 173361061                   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The condition variable's p-value is low (indicated by the '***'), implying that the difference in the average home price influences the mean salary. As a result, H0 is rejected. The average salary of software developers varies significantly depending on the mean accommodation expenses in each metropolitan area.

Tukey’HSD test to determine the difference between each two segments of the figure for average home price:

```
                         diff        lwr      upr     p adj
> 150000-<1000      15350.525  -7467.287 38168.34 0.2482386
1000-2000-<1000      9484.239   2195.356 16773.12 0.0073586
1000-2000-> 150000 -5866.286 -28761.578 17029.01 0.8135809
```

There are statistically significant differences ($p < 0.05$) between segment groups: 1000-2000 group and below 1000 group. The other group differences are not statistically significant.

We continue to use the Levene test to verify the null hypothesis that the population variances are equal

```
> leveneTest(unadjusted_dev_salary ~ as.factor(avg_home_price), data = sD)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value  Pr(>F)  
group  2  3.0533 0.05319 .
      74                  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Calculating the tests yields a p value of 0.05319, which is greater than the 5% level of significance. As a result, there is no reason to reject the null hypothesis H0. We can say that the variance between groups appears to be equal.

To make the result more clear, we will apply the Shapiro-Wilk test to assess the normality assumption based on the residuals.

```
> aov_residuals_1 <- residuals(object = one_way_aov_1 )
> shapiro.test(x = aov_residuals_1 )

        Shapiro-Wilk normality test

data:  aov_residuals_1
W = 0.94418, p-value = 0.002063
```

The p-value is approximately zero, thus we can reject H0 and state that the data is non-normal at high certainty.

4.3.4. Linear regression analysis

To predict job growth for software developers by analyzing factors like local purchasing power, median home price and number of jobs available, the author group apply linear progression method to create a model. Consider a linear regression model with the variable unadjusted_dev_salary as the dependent variable and num_dev_jobs, avg_home_price, living_expenses, avg_purchasing_power as independent variables. To run a multiple linear regression model, use the **lm(...)** command:

- First step is to randomly split the whole dataset into training data and testing data. Test dataset purpose is to test the model trained with training dataset. We use 70% of the dataset as training set and 30% as test set.

- Then we create a standard linear regression model with the **lm(...)** command. The complexity or the accuracy of this linear model is relatively low since it is a simple linear regression.

```
Call:
lm(formula = unadjusted_dev_salary ~ num_dev_jobs + avg_home_price +
    living_expenses + avg_purchasing_power, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-13470.3  -3483.4   -172.3   3975.8  11536.3

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          8.517e+04  5.887e+03  14.467  < 2e-16 ***
num_dev_jobs         2.730e-01  5.596e-02   4.879 1.17e-05 ***
avg_home_price       3.028e-02  5.261e-03   5.756 5.55e-07 ***
living_expenses      2.942e-02  5.062e-02   0.581    0.564
avg_purchasing_power 2.387e-01  7.874e-01   0.303    0.763
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6310 on 49 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.8139,    Adjusted R-squared:  0.7987
F-statistic: 53.57 on 4 and 49 DF,  p-value: < 2.2e-16
```

As shown parameters of lm() function, we are trying to find the right relation between unadjusted_dev_salary with other factors. In other words, the formula of developer salaries can be described as:

unadjusted_dev_salary = (num_dev_jobs * c1) + (avg_home_price * c2) + (living_expenses * c3) + (avg_purchasing_power * c4) + c5

With: c1 = 2.73*(10^-1); c2 = 3.028*(10^-2); c3 = 2.942*(10^-2); c4 = 2.387*(10^-1); c5 = 8.517*(10^4)

The residuals, or the difference between our fitted model predictions and the unadjusted developers salary, range from -13470.3 to 11536.3, with a median of -172.3. The residuals in our case are not symmetrical. The 1Q value matches up to the 3Q value, except the maximum and minimum residuals are severely skewed. This information indicates that linear regression does not fit this data too well, or that there are many outliers in the price data affecting the model's ability to fit. The coefficients of our linear regression shows the relation between the price and other factors. Each coefficient has a rather low magnitude of t-value. and > 0.05 and therefore statistically significant. With the null hypothesis of there being no relationship between house price and other factors, we can reject the null hypothesis and prove that there exists a relationship between house price and others. The value of Pr(> |t|) calculated shows that living expenses and average purchasing power are statistically significant, while the number of jobs and house prices are not statistically significant.

This model has Multiple R-Squared value of 0.8139, therefore the percentage of variance in developer salaries can be explained by the data is at an acceptable level.

4.4. Summary

First, using R's built-in functions, we found out that 'Columbus, OH' is the city with the best compensation package, while `Honolulu, HI' being the worst.

Then, by applying the established total cost of relocating formula, the resulting plot shows that relocating to California would be the most expensive, followed by New York and then Texas.

To predict job growth for software developers, we used Tukey's HSD to judge the proposed hypotheses, and we concluded that factors including local purchasing power, median home price, and number of available jobs have a strong relationship with developer's salary. And to ascertain that population variances are equal, we used Levene and Shapiro-Wilk test.

Finally, with the multiple R squared value of 0.8139, we confirmed that the Linear regression model is an acceptable model for predicting job growth for software developers.

5. Discussion and extensions

We have applied a linear regression model in the previous section to predict job growth for software developers by analyzing the following factors: local purchasing power, median home price, living expenses, and number of jobs. The obtained R squared value indicates that the relationship between the dependent and independent variable is somewhat linear.

In this section, we will instead apply the polynomial regression model, a machine learning model that captures nonlinear relationships between variables by fitting a non-linear regression line, using Python. Our target here is unadjusted_dev_salary, instead of adjested_dev_salary, and with the same features as in the linear regression session.

In polynomial regression, the relationship between the dependent variable and the independent variable is modeled as an nth-degree polynomial function. When the polynomial is of degree 2, it is called a quadratic model; when the degree of a polynomial is 3, it is called a cubic model, and so on. We will assume the the polynomial function has a degree of 2, 3 or 4, and then compare the resulting R squared values

First, let's import the necessary libraries.

```
# Libraries
import pandas
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

Then, we read the data.

```
# Target and features
features = ['num_dev_jobs', 'avg_home_price', 'living_expenses', 'avg_purchasing_power']
target = ['adjusted_dev_salary']

# read dataset
df = pandas.read_csv('dataset.csv')

# Only use target and features column
X = df.loc[:, features]
y = df.loc[:, target]
```

We define our function of the polynomial regression model, where n is the degree of the polynomial function.

```
def poly_regression_model(n):
    # Create the new polynomial features
    poly = PolynomialFeatures(degree=n, include_bias=False)
    X_poly = poly.fit_transform(X)

    # Split the dataset into train and test using train_test_split function
    X_train, X_test, y_train, y_test = train_test_split(X_poly, y, random_state=0, train_size=.8)

    # Save an instance of LinearRegression() to pr
    pr = LinearRegression()

    # Fit our model to pr
    pr.fit(X_train, y_train)

    # Print R squared value
    print(f"R^2 of polynomial regression model with degree of {n}: {pr.score(X_train, y_train)}")
```

To compare results from linear regression models, we define a linear regression model function.

```python
def linear_regression_model():
    # Split the dataset into train and test using train_test_split function
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, train_size=.8)

    # Save an instance of LinearRegression() to lr
    lr = LinearRegression()

    # Fit our model to lr
    lr.fit(X_train, y_train)

    # Print R squared value
    print(f"R^2 of linear regression model: {lr.score(X_train, y_train)}")
```

Finally, here's the output after we call the functions.

```python
linear_regression_model()
poly_regression_model(2)
poly_regression_model(3)
poly_regression_model(4)
```

```
R^2 of linear regression model: 0.17076948626151423
R^2 of polynomial regression model with degree of 2: 0.4017305289193457
R^2 of polynomial regression model with degree of 3: 0.6366210962215636
R^2 of polynomial regression model with degree of 4: 0.3276161666324129
```

Judging from the results, the polynomial regression model with the degree of 3 yields the highest R squared value among the three degrees, and performs approximately 3.7 times better than the linear regression model. With the R squared value of roughly 0.637, The polynomial regression model shows the independent and dependent variables have moderate relationship, and is a relatively promising model for this dataset.

6. Code and Data

- Data: U.S. Software Developer Salaries | Kaggle

- Code: Code

7. References

[1] Cost of living calculation: https://www.investopedia.com/ask/answers/100214/how-cost-living-index-calculated.asp, https://www.nerdwallet.com/cost-of-living-calculator

[2] Analyticsvidhya.com's linear regression blogs: https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-

know-about-linear-regression/, https://www.analyticsvidhya.com/blog/2022/01/different-types-of-regression-models/

[3] BioSTTS. (n.d.). Post hoc tests - tukey HSD. Retrieved November 21, 2022: https://biostats.w.uib.no/post-hoc-tests-tukey-hsd/

[4] Fitting Linear Models in R - Rdocumentation: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm

[5] One-way ANOVA test in R: http://www.sthda.com/english/wiki/one-way-anova-test-in-r

[6] Introduction to Polynomial Regression Analysis: https://serokell.io/blog/polynomial-regression-analysis

[7] Multiple Linear Regression: https://corporatefinanceinstitute.com/resources/data-science/multiple-linear-regression/