Tokaeva Aleksandra – HW2

1. Background

I chose the dataset from here https://www.kaggle.com/datasets/esratmaria/house-price-dataset-with-other-information; It contains information about 21613 houses in King County, USA, described by 21 parameters (=features); we will try to predict the price of a house (it is a real-number) using other features of this house; to do this, we will first do data preprocessing (to remove outliers), and then use linear prediction, linear regression and knn method, described in the lectures. And then we will make conclusions.

2. Dataset

Dataset has the shape (21613,21); each row is a house, described by 21 parameters ((id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, sqft_above, sqft_basement, yr_built, yr_renovated, Zipcode, lat, long, sqft_living15, sqft_lot15); id is just some code, it is useless; price is the price of the house; bedrooms is the amount of bedrooms; bathrooms is the amount of bathrooms; sqft_living is the living area; sqft_lot is the total area; waterfront is the binary feature, telling if the house faces the waterfront; condition and grade are category features; sqft_above, sqft_basement are the area of house above the ground and under the ground respectively; yr_built, yr_renovated are years if construction and renovation; lat, long are the latitude and longitude; sqft_living15, sqft_lot15 are the area of 15 nearest neighbors;

We remove columns id, zipcode – because they have no influence on the price of the house; at and long may have impact on price – but to understand it one needs to know the area (for example, if some houses are near to the factory and hence have dirty air – they will cost less, but one needs to know the fact of the factory presence in the area); we do not have the knowledge of the area, so we will remove the lat and long;

3. Analysis

First, we need to see if there are any outliers – otherwise we will have difficulties with model performance, because if thresh in – then thresh will be out; So we build the plt.scatter dependence of price with respect to each feature; also we could use sns.boxplot(y='price', x='bedrooms', data=df); we see that one house has 33 berdrooms; it is definitely an outlier- we remove it; we also remove houses with sqrt_living > 9000 - they are outliers; remove houses with sqft_lot > 600000; houses with sqft_above > 7000; remove houses with sqft_basement > 3000; remove houses with sqft_lot15 > 400000; and we see that features sqft_lot and sqft_lot15 are of different order than others => we need to take logarithms of them; and we also see the problem with yr_renovated - it is zero for some houses; let's interpret it as if they were never renovated - and replace yr_renovated with yr_built; after that we plot the correlation matrix using sns.heatmap(df.corr(), cmap = 'viridis',annot = True) – to remove correlatated features; it is both because they carry no new information, and also it is dimentionality reduction, and also the exact formula for linear regression has problems, because the matrix will be degenerate – so better to remove correlated features; we see that yr_built and yr_restructed are correlated with coefficient 0.91 – so we will remove yr_built;

Then we will split the data on train and test, in proportion 7:3, using train_test_split from sklearn.model_selection because we need some data to score the results of our model – and we can't score the model on train sample;

First of all, lets try to fit linear regression on price vs sqft_living - from lecture we know the exact formulas for the coefficients k0 and k1; we use them to plot the red line; and we also compute on train such metrics as mae, mape, mse; so for 1-factor linear regreassion the mape score is 36% - it is a big error; now we will try to improve it using multi-factor linear regression and knn regression;

Then we use LinearRegression() from sklearn.linear_model and improve the Mape score from 36% to 31%; also the LinearRegression() is useful because it can show the weights of the feactures – in our case, the largest weights have features sqft_living/basement/above, waterfront and grade – and the fact that these features have great impact on the price of the house was evident from the beginning – and this signals that our linear regression model is not senseless, which is good;

Finally, we use KNeighborsRegressor from sklearn.neighbors to use k nearest houses from train to predict the price of a new house from test sample; we explore the Mape score for k from 10 to 100 and choose k that gives the least mape – it is k=18; surprisingly, the rule of thumb k~sqrt(N) is not the best choice in our case; with k=18 we have Mape=31%, which is better than 1-factor linear regression, but worse than multi-factor linear regression;

4.Conclusion

To sum up, we have explored how the price of the house in King County depends on the features of the house; first, we removed outliers from the dataset; then, we removed the correlated features; and then we used three approaches to predict the price of the house having 21 features of this house; we saw that the dependence of price from sqft_living (the most obvious feature) is not really linear and the sqft_living alone is not sufficient to predict the price – the mape error is 36%; then we reduced the mape error to 29% using linear regression – and also understood that the features with largest weight are sqft_living/basement/above, waterfront and grade; and finally we used knn regressor model than uses the price of the k kearest neighbors to predict the price of a new house – but the mape error is 32%; further approaches could be to train a neural network on these 21 features and see what mape error we will have;

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0. Our goal is to predict the price of the flat using its features (e.g. its sqft_living and etc.); the data is from here https://www.kaggle.com/datasets/esratmaria/price-dataset-with-other-information (https://www.kaggle.com/datasets/esratmaria/price-dataset-with-other-information)

In [2]:
```
df = pd.read_csv('kc_house_data.csv')
print(df.shape) #we see that there are 21613 houses each having 21
df.head(5)
```

```
(21613, 21)
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1

5 rows × 21 columns

1. Let's Remove 'id','zipcode','lat','long' - because they are not useful; and replace date by year

In [3]:
```
df=df.drop(columns=['id','zipcode','lat','long'],axis=1)
print(df.shape)
df.head(5)
```

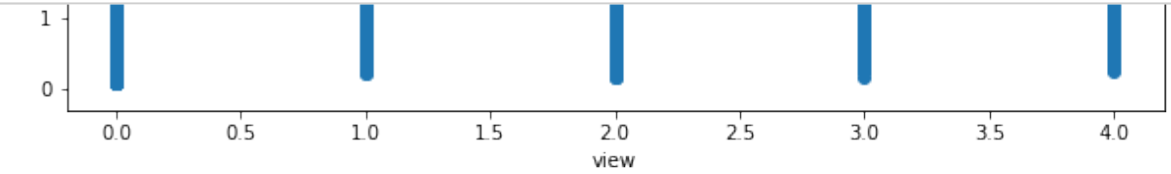
```
(21613, 17)
```

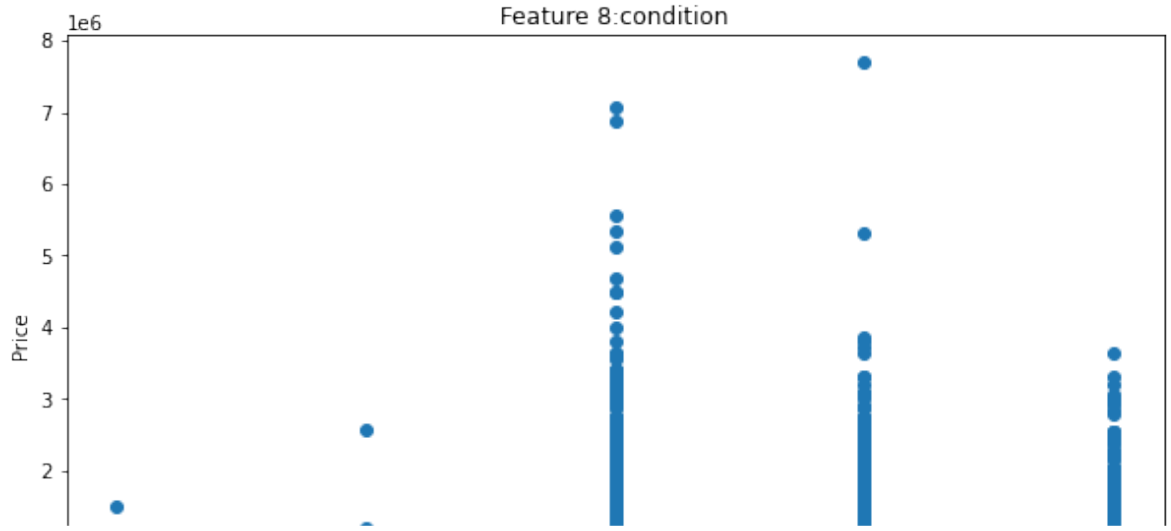
Out[3]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfron
0	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

2. Let's analyse the columns - may be there are some outliers that we need to remove

In [4]:
```
fig=plt.figure(figsize=(10,100))
ax=fig.subplots(df.shape[1]-2,1)
for i in range(2,df.shape[1]):
    col_name=df.columns[i]
    ax[i-2].scatter(df[col_name],df['price'])
    ax[i-2].set_xlabel(col_name)
    ax[i-2].set_ylabel('Price')
    ax[i-2].set_title('Feature {}:{}'.format(i-1, col_name))
plt.show()
```





In [5]:
```
### Analysis: we see a definite outlier with bedrooms - a house wit
### Also let's remove houses with sqrt_living > 9000 - they are out
### Also remove houses with sqft_lot > 600000
### Also remove houses with sqft_above > 7000
### Also remove houses with sqft_basement > 3000
### Also remove houses with sqft_lot15 > 400000

df=df[df['bedrooms']<=15]
df=df[df['sqft_living']<=9000]
df=df[df['sqft_lot']<=600000]
df=df[df['sqft_above']<=7000]
df=df[df['sqft_basement']<=3000]
df=df[df['sqft_lot15']<=400000]
len(df)
```

Out[5]: 21581

In [6]:
```
### And we definitely see the problem with yr_renovated - it is zer
df.loc[df['yr_renovated']==0, ['yr_renovated']]=df['yr_built']
df['yr_built']=df['date'].str[0:4].astype(int)-df['yr_built']
df['yr_renovated']=df['date'].str[0:4].astype(int)-df['yr_renovated
df=df.drop('date', axis=1)
df.head(5)
```

Out[6]:

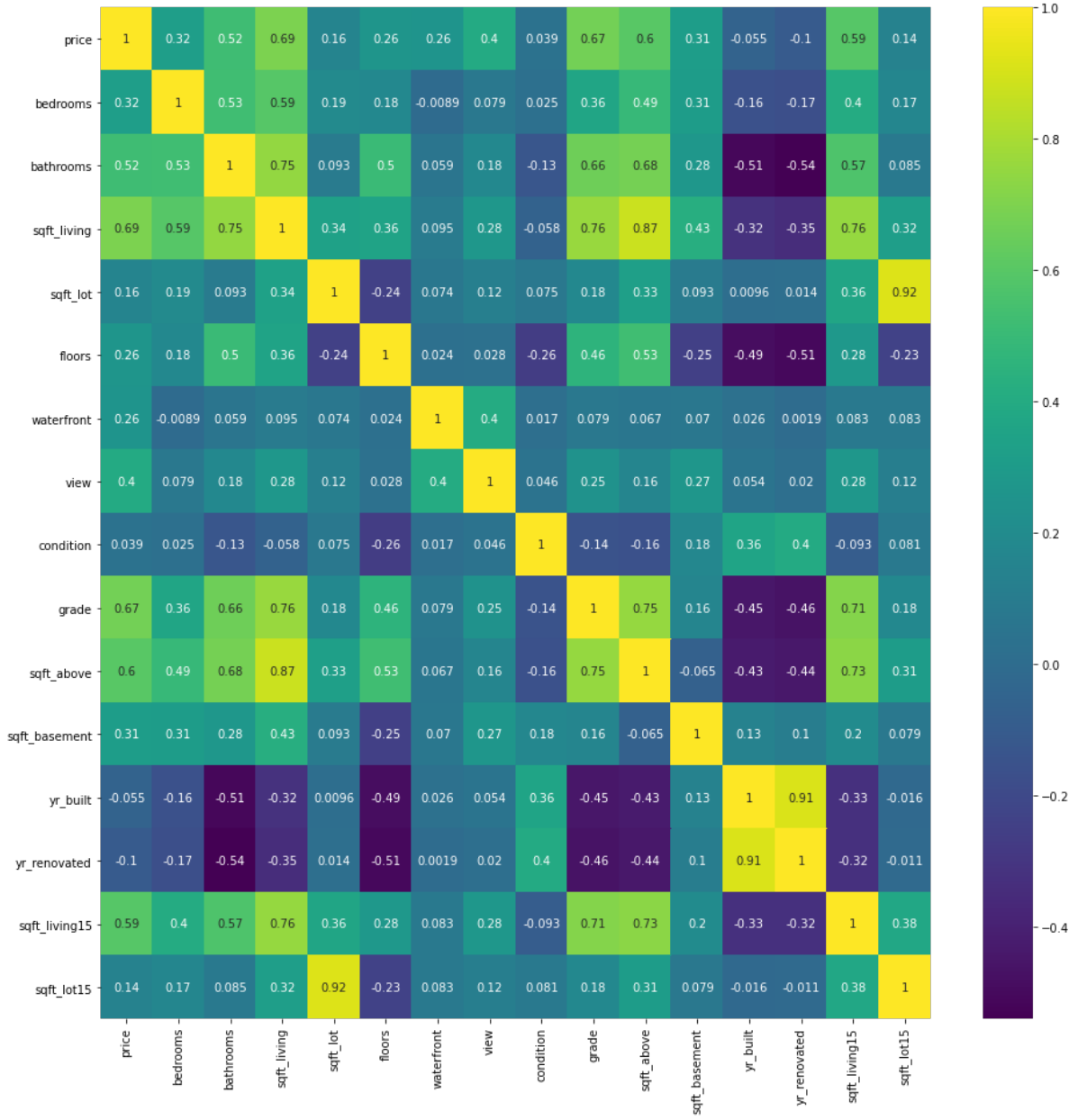
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	221900.0	3	1.00	1180	5650	1.0	0	0	3
1	538000.0	3	2.25	2570	7242	2.0	0	0	3
2	180000.0	2	1.00	770	10000	1.0	0	0	3
3	604000.0	4	3.00	1960	5000	1.0	0	0	5
4	510000.0	3	2.00	1680	8080	1.0	0	0	3

In [7]:
```
### and also we see that features sqft_lot and sqft_lot15 are of di

df['sqft_lot']=np.log(df['sqft_lot'])
df['sqft_lot15']=np.log(df['sqft_lot15'])
```

3. Now see if the features are highly corrlated - and remove those that are corralated => we will remove yr_built

In [8]:
```
plt.figure(figsize=(16, 16))
sns.heatmap(df.corr(),  cmap = 'viridis',annot = True)
plt.show()
```



In [9]:
```
df=df.drop('yr_built', axis=1)
```

4. Now split our data to train and test

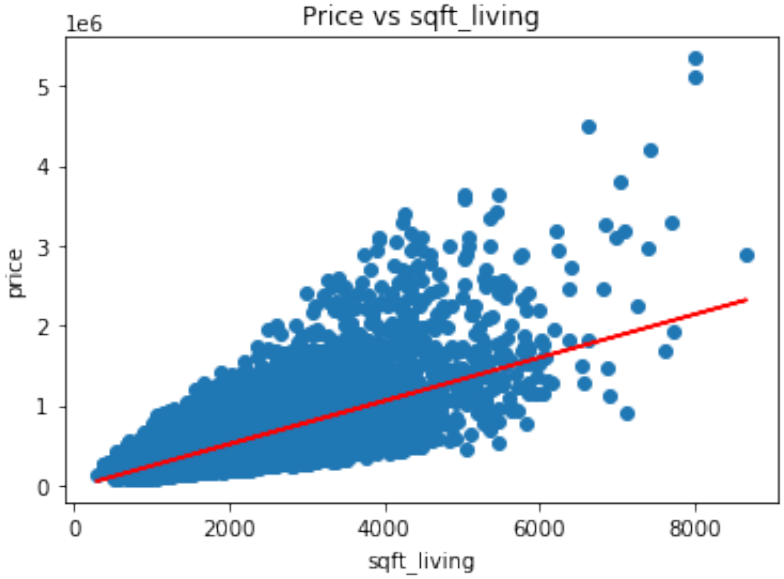
In [10]:
```
Y=df['price']
X=df.drop ('price',axis=1)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test  = train_test_split(X, Y, test_siz
```

5. Lets try to fit linear regression on price vs sqft_living - from lecture we know the exact formula for the coefficients k0 and k1

$$k_0 = \frac{<y><x^2>-<x><xy>}{<x^2>-<x>^2} \; k_1 = \frac{<yx>-<y><x>}{<x^2>-<x>^2}$$

In [11]:
```
x=X_train['sqft_living'].values
y=Y_train.values
k0=(y.mean()*(x*x).mean() - x.mean()*(x*y).mean())/((x*x).mean() -
k1=((y*x).mean() - y.mean()*x.mean())/((x*x).mean() - (x.mean())**2
print("k0=",k0,"k1=",k1)
plt.scatter(x,y,label='data')
plt.plot(x,[k0+k1*xx for xx in x],color='red',label='best linear fi
plt.xlabel('sqft_living')
plt.ylabel('price')
plt.title('Price vs sqft_living') #we see that it is not a very goo
plt.show()
```

```
k0= -23740.563830769206 k1= 270.81754403284987
```



In [12]:
```
# we measure the mape error
from sklearn.metrics import mean_absolute_error, mean_squared_error
x_test=X_test['sqft_living'].values
y_test=Y_test.values
y_pred=k0+x_test*k1

def MAPE(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

print ('MAPE:', MAPE(y_test, y_pred)) #we see that error is too big
print ('MAE:', mean_absolute_error(y_test, y_pred))
print ('√MSE:', mean_squared_error(y_test, y_pred)**(1/2))
print ('R2_score:', r2_score(y_test, y_pred))
```

```
MAPE: 35.99401273142257
MAE: 171481.93068511266
√MSE: 257653.4098650002
R2_score: 0.4793607932105457
```

6. Let's try fit Linear regression on all features - error should become less

In [13]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, Y_train)
y_pred = model.predict(X_test)

print ('MAPE:', MAPE(y_test, y_pred)) #we see that error is better
print ('MAE:', mean_absolute_error(y_test, y_pred))
print ('√MSE:', mean_squared_error(y_test, y_pred)**(1/2))
print ('R2_score:', r2_score(y_test, y_pred))
```

```
MAPE: 28.92641288357411
MAE: 137669.04177317797
√MSE: 212582.2695647874
R2_score: 0.6455792343535439
```

In [14]:
```
# lets look on what features have more weight => we see that sqft_l
print("Here are the features with maximum weights:")
sorted(list(el for el in zip([abs(x) for x in model.coef_], X_train
```

```
Here are the features with maximum weights:
```

Out[14]:
```
[(555978.17547727, 'waterfront'),
 (114643.31358626508, 'grade'),
 (48288.324514865635, 'view'),
 (33257.28343731093, 'bedrooms'),
 (31748.324413904284, 'bathrooms')]
```

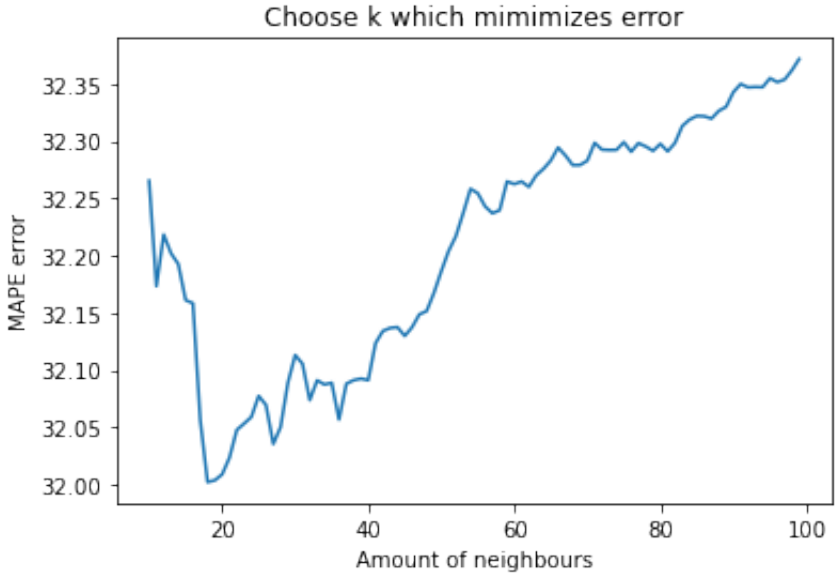
7. Let's try knn algorithm to predict price - and choose k from 1 to len(X_train)

In [15]:
```
from sklearn.neighbors import KNeighborsRegressor
from tqdm import tqdm
mas_errors=[]
N=100
for k in tqdm(range(10,N)):
    model = KNeighborsRegressor(n_neighbors=k)
    model.fit(X_train, Y_train)
    y_pred=model.predict(X_test)
    tek_error=MAPE(y_test, y_pred)
    mas_errors.append(tek_error)

print("Best k is:", np.argmin(mas_errors)+10)
plt.plot(range(10,N), mas_errors)
plt.xlabel('Amount of neighbours')
plt.ylabel('MAPE error')
plt.title('Choose k which mimimizes error')
plt.show()
```

```
100%|██████████| 90/90 [00:09<00:00,  9.14it/s]

Best k is: 18
```



In [18]:
```
model = KNeighborsRegressor(n_neighbors=18)
model.fit(X_train, Y_train)
y_pred=model.predict(X_test)
print ('MAPE:', MAPE(y_test, y_pred)) #we see that error is better
print ('MAE:', mean_absolute_error(y_test, y_pred))
print ('√MSE:', mean_squared_error(y_test, y_pred)**(1/2))
print ('R2_score:', r2_score(y_test, y_pred))
```

```
MAPE: 32.002095059543905
MAE: 154910.98168168167
√MSE: 237807.47004272044
R2_score: 0.5564770819013818
```

In [17]:
```
# we see that knn works but also not very well
```

In [ ]:
```

```

Name: Alexsandra Tokaeva

Methodology: The student used the techniques including data preprocessing, feature selection based on correlation, linear regression and kNN methods for regression. Data are well-explained and processed. The methods are well implemented. I particularly like the heatmap you plotted for feature selection. However, methods are not well-explained. For example, when knn are applied, what loss function are used in your project? This is unclear to the reader.

25/30

Content: Overall, it is a good report on predicting house prices based on regression. The dataset was chosen from Kaggle which includes the house price with 21 features. 14 features are kept after data preprocessing and feature selections. The student spent most parts explaining how data was processed and what functions are used from python. More space can be used for giving motivations, method descriptions and discussions on the results, instead of focusing too much on the technical details such as which python command is used. For example, kNN regression gave worse MAPE error than linear regression. What is the possible issue?

30/40

Presentation: The structure of this report is clear. There are a few things that may need to draw attention to. 1. The report needs a title; 2. The layout of the report. Sections are not aligned. Fonts are all the same size; 3. Avoid using informal languages and '-'; 4. Be careful when you want to mention programming languages in your main report. Instead, just mention the name of method or the type of plot. 5. Your codes and plots are not fully displayed.

20/30

Overall: *75%*