Question earnings prediction and analysis on the website Quora

1st, February 2022

I. Introduction

In this report, I will describe my attempt at evaluating how much money can be earned from each posted question on the website Quora.

Quora is a social question-and-answer website based in California, United States. It is considered to be one of the biggest question-and-answer applications/websites in the world, along with others such as Answers, StackExchange and Quizlet.

To increase the traffic volume and search appearance on Google, Quora needs to host an immense number of different questions spanning all topics. The intuition is that any internet user will see their concerned questions posted on Quora, leading to their clicking on the link and visiting the website. In question-answer websites like Quora, users are noted to be more likely to answer existing questions rather than post a new question. To significantly increase the number of questions, Quora has devised a strategy called Partner Program in 2018, where users in this program are paid for regularly posting new questions, helping to increase the website's traffic volume.

Source of the data: The question data is directly based on my own posted questions on Quora, which I have requested Quora to send me a copy of the data. My Partner Program was in the Japanese language, so this report paper will only investigate the income scheme of the program in Japanese, which is probably inapplicable to programs in other languages.

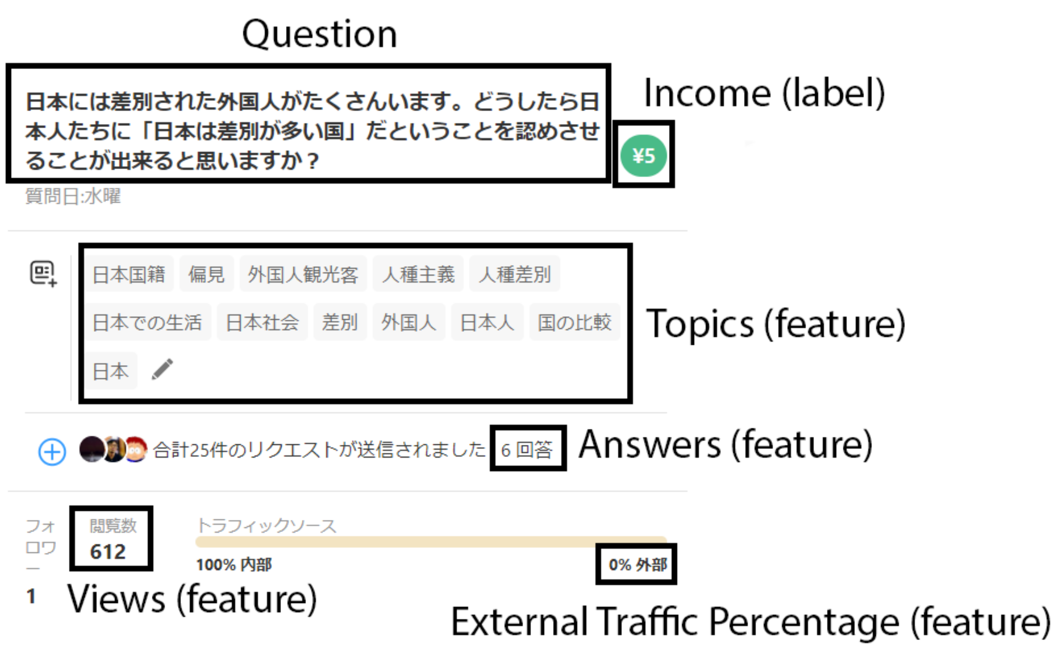
Below this, I will discuss several machine learning methods for analyzing the correlations between the individual questions' income and their features. Finally, I will make predictions and provide a conclusion on the strategy to maximize the income from the Quora Partner Program.

II. Problem Formulation

The question that I am trying to answer is: given some certain characteristics of the posted questions, how much income the question is likely to earn?

According to official information, the question's income is based on how often external users click on the question, stay on the website for a certain duration of time and see the advertisements promoted by Quora. However, based on my experience, it is particularly difficult to predict whether the question will earn much money or not generate any income. This is the purpose of this paper: to systematically gather the necessary data and make a general analysis and prediction towards the income scheme of the Quora Partner Program.

For each question datapoint, various features can be recorded, but I opt to choose only four features because they are easy to be collected in a highly automated manner, namely: the number of answers, the number of views, the external traffic percentage and the question topics. The illustration of the datapoint is clearly illustrated in Fig 1.



日本には差別された外国人がたくさんいます。どうしたら日本人たちに「日本は差別が多い国」だということを認めさせることが出来ると思いますか？

質問日:水曜

日本国籍 偏見 外国人観光客 人種主義 人種差別 日本での生活 日本社会 差別 外国人 日本人 国の比較 日本

合計25件のリクエストが送信されました 6 回答

フォロワー 1 閲覧数 612 トラフィックソース 100% 内部 0% 外部

Figure 1: A posted question on the Quora website with many different features

Summary of the machine learning problem

Datapoint: A single question with five properties: number of answers, number of views, external traffic percentage, question topics, and the generated income.

Features: The number of answers made to the question, the number of views when users click on the question, external traffic percentage and the topics of the question.

Label: Income of the question (in Japanese yen). The income information is not available when the question is posted. It is only known the day after the question posting => It is uncertain how much the question will earn at the time of posting.

III. Methods

3.1 Dataset

From the synopsis above, the data points represent information of a single posted question. Features and label of the datapoints are explained in problem formulation part

The datatype of the features and labels:

- Features: number of answers, the number of views, external traffic percentage are integer
- Feature topics of the question is string
- Label income is integer

Although there are over 7000 questions I have posted, I only choose the top 500 earning questions to study their properties and patterns. Therefore this dataset has 500 datapoints, which should suffice for question selections. There is no missing data in any fields.

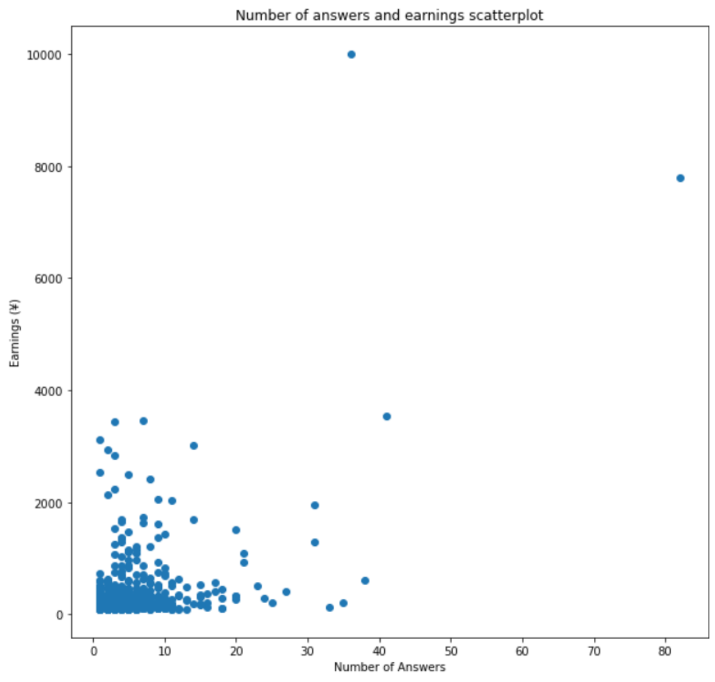
To collect the data points for this project, I edit the exported file that Quora has sent me so it could be readable by Python libraries, as well as manually adding some features like question topics and external

traffic percentage. Therefore, the dataset is not supposed to be found publicly as the income data in this report is strictly exclusive to me, but nevertheless you can verify them at the URL provided below.
https://jp.quora.com/profile/Nguyen-Thanh-Luan

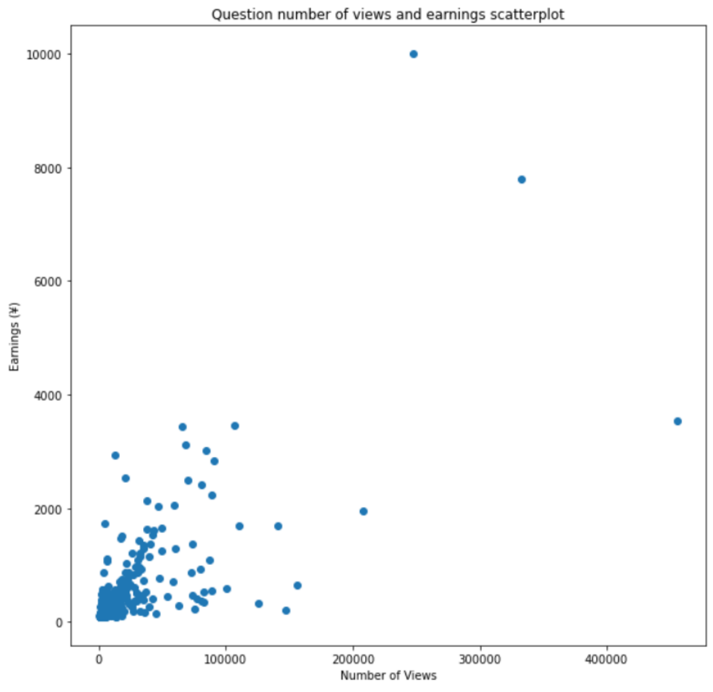
3.2 Feature Selection

First of all, I plot some data visualizations between the earnings label and the individual features

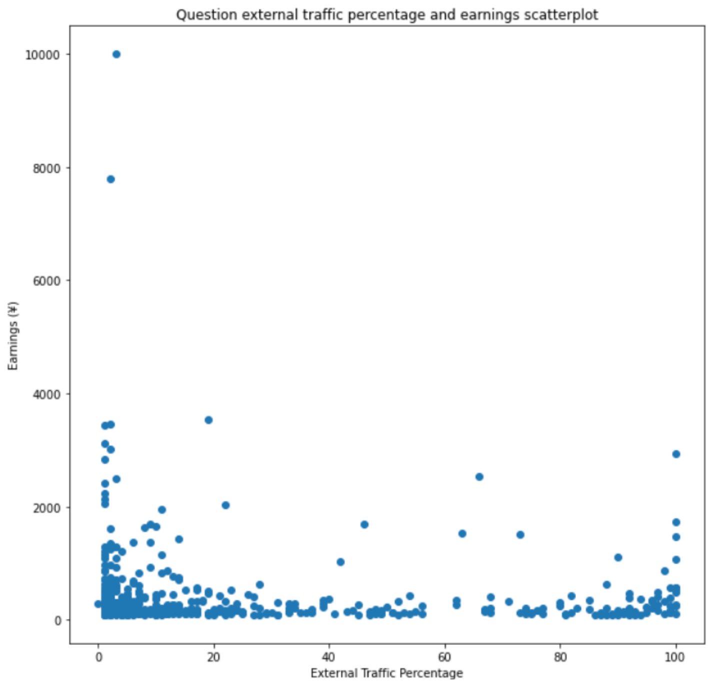
- Question number of answers and earnings scatterplot



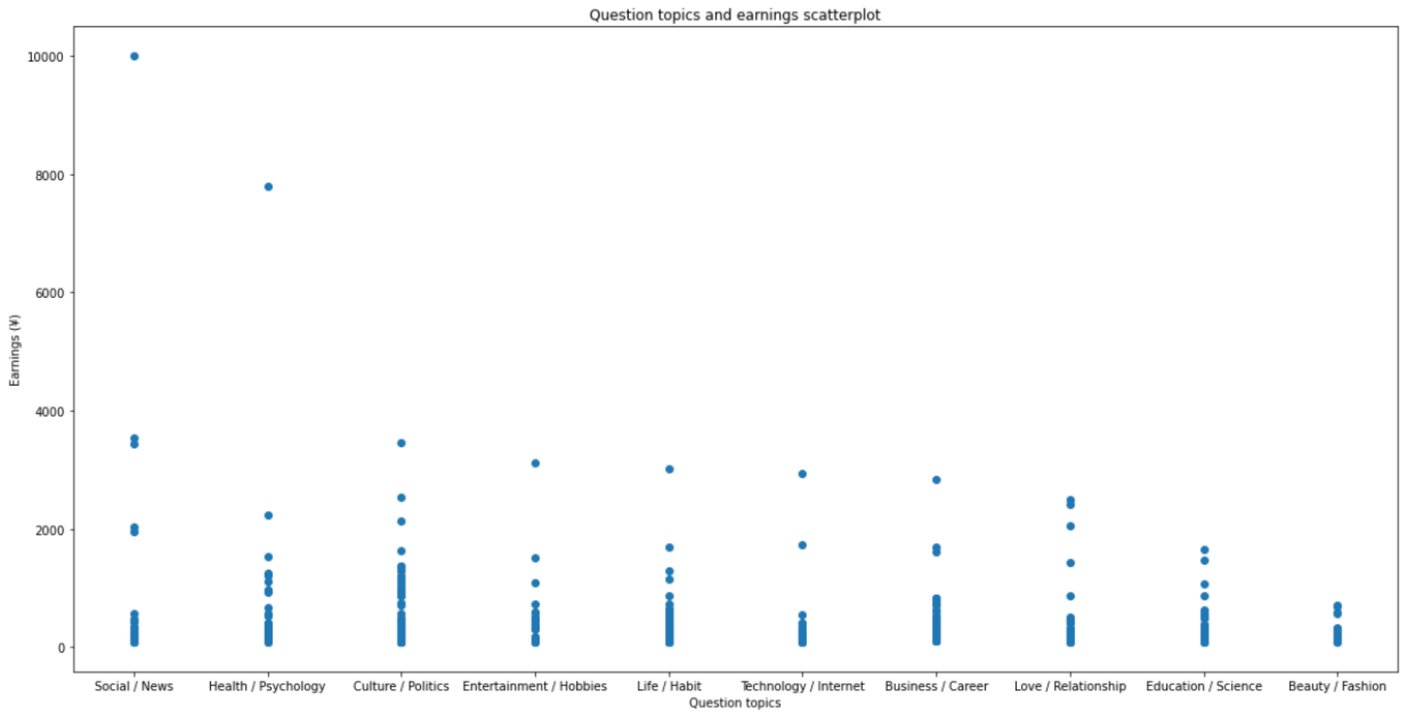
- Question number of views and earnings scatterplot



- Question external traffic percentage and earnings scatterplot



- Question topics and earnings scatterplot



From the scatterplots, there seems to be a linear relationship between the feature views, answers and label earnings, while there are no clear relations between the income and external traffic percentage. The topics also correlate with earnings, because there are certain topics that are likely to earn more than the others.
=> Conclusion: The features answers, views and topics are chosen for this ML problem and the feature external traffic percentage is discarded for this project.

3.3 Model (Hypothesis Space) Selection & Loss functions

a) Model Selection

Due to the nature of the data points, I will separate the model selections into two types: the linear method and the classification method. The justifications are as follows

- The linear method will be used for modeling the features number of views and number of answers, since in the data plots, they appear to be linearly related to the income. The two linear methods I will discuss are namely the Huber Regression and the Ridge Regression.
- The classification method will be used for modeling the question topics. For question topics, it contains 10 discrete topics and thus it is sensible to use classification methods for this feature. The two classification methods I will discuss are logistic regression and support vector machine (SVM), respectively.

The hypothesis space of the four methods are:

1. Huber regression: Linear maps
2. Ridge regression: Linear maps
3. Logistic regression: Linear maps
4. Support vector machine: Linear maps

b) Loss function

The following loss functions will be chosen for their respective model selections are:

- Huber Regression: The Huber loss functions [1]

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \hat{y}_i)^2 \qquad |y_i - \hat{y}_i| \le \delta$$

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) \qquad |y_i - \hat{y}_i| > \delta$$

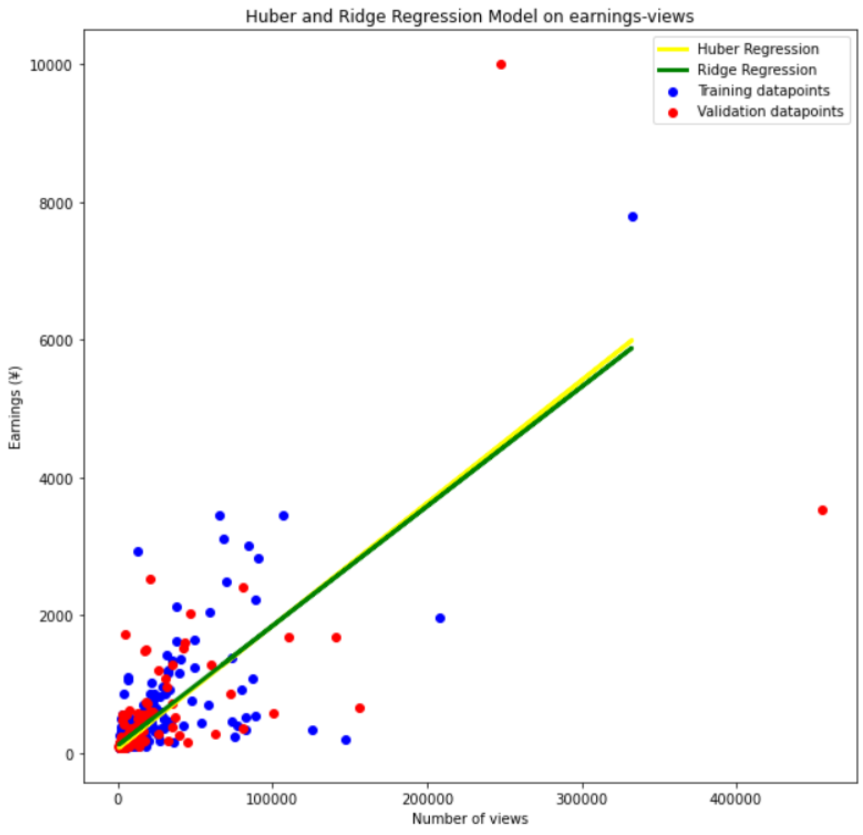
The reason Huber regression and loss function are considered for the feature answers and views is because there are a few extreme outliers in the dataset that will severely affect the result of the regression (Some questions have abnormally larger earnings than most of the rest questions). The Huber loss function is robust against outliers and thus provides a better prediction than a normal Linear Regression.

- Ridge Regression: MSE loss with L2-norm regularization [2]

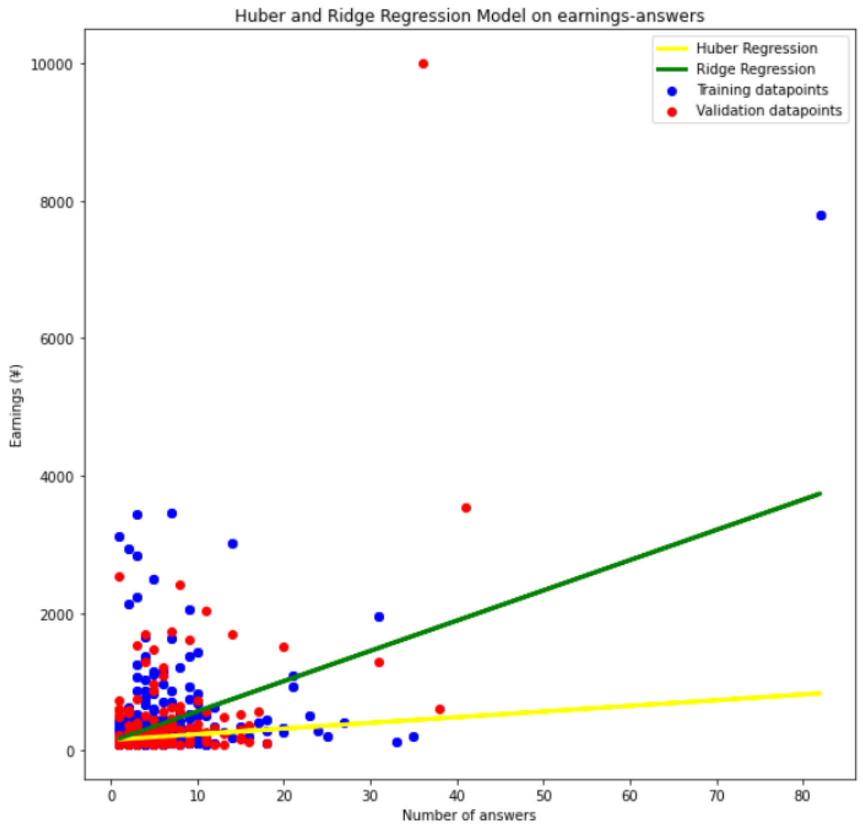
$$\begin{aligned}\hat{\beta}^{\text{ridge}} &= \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \\ &= \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}\end{aligned}$$

Given the visualizations above, the datapoints have considerably large variance and the normal Linear Regression is expected to perform badly on validation sets. The Ridge Regression trades off an increase in bias with a decrease in variance so that the predictor would yield better results. The bias is increased by introducing a penalty term that is the product of regression slope and the hyperparameter λ

The plots below show the regression of these two models on the two features answers and views



Huber and Ridge Regression Model on earnings-views
Huber Regression
Ridge Regression
Training datapoints
Validation datapoints
Earnings (¥)
Number of views
0
2000
4000
6000
8000
10000
0
100000
200000
300000
400000



Huber and Ridge Regression Model on earnings-answers
Huber Regression
Ridge Regression
Training datapoints
Validation datapoints
Earnings (¥)
Number of answers
0
2000
4000
6000
8000
10000
0
10
20
30
40
50
60
70
80

For the feature topics, the loss functions of the two classification models are

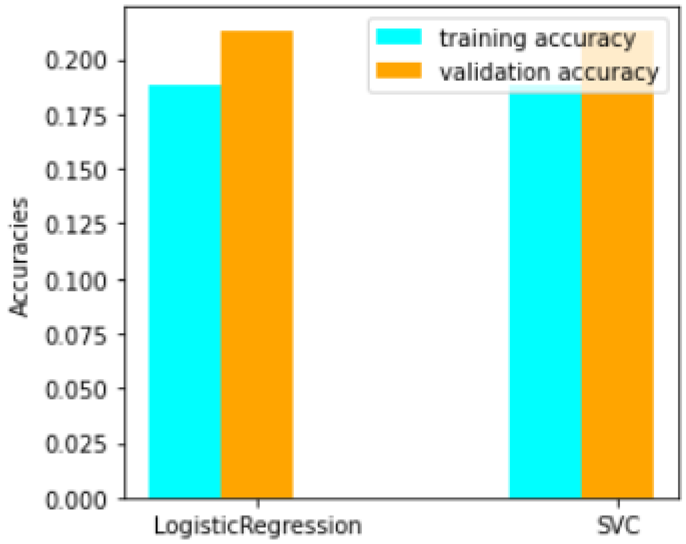
- Logistic Regression: Logistic loss

Logistic loss indicates how close the prediction probability is to the corresponding actual value or label and as the probability of prediction for label becomes lower and further from the actual label, the more the logistic loss increases. Particularly, logistic loss is sensitive to outliers and there are many outliers in my dataset, which makes this loss function a good testing one for my model.

- Support vector machine: Hinge loss

The hinge loss incorporates a margin away from the classification boundary into the cost calculation. Even if new observations are classified correctly, a penalty can be imposed if the margin from the decision boundary is not large enough. [4] Since in my dataset, there are quite many classes (10 topics) and the boundaries are not very transparent. The hinge loss will thus penalize small margin decisions.

- Preliminary results of the two classification models on earnings - topics



3.5 Training and Validation Set

In this project, I would use a simple training test split that splits the datapoints into the training and validation set. Current consensus for the ratio of training/validation size for very large datasets are 80/20% to 90/10% and for small dimensional datasets are 60/40% to 70/30% [3]. Therefore, I would choose the ratio 70/30 % for my project. Also, I believe that 70/30% ratio is ideal for my dataset because there is great variance in the features, so this proportion is ideal to ensure that the model does not overfit the training data

IV. References

[1] Calculate Huber Loss using TensorFlow 2, available at
https://lindevs.com/calculate-huber-loss-using-tensorflow-2/
[2] Ridge Regression and Lasso Regression, available at
https://discuss.boardinfinity.com/t/ridge-regression-and-lasso-regression/6208
[3] Available at
https://www.researchgate.net/post/Is-there-an-ideal-ratio-between-a-training-set-and-validation-set-Which-trade-off-would-you-suggest
[4] Understanding Hinge Loss and the SVM Cost Function. Available at
https://programmathically.com/understanding-hinge-loss-and-the-svm-cost-function/

V. Code appendix

```
# %config Completer.use_jedi = False  # enable code auto-completion
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns  #data visualization library
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix  # evaluation metrics
from sklearn.linear_model import HuberRegressor, LinearRegression, LogisticRegression
from sklearn.model_selection import KFold
from sklearn.neighbors import NearestNeighbors
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split, KFold
from sklearn.svm import SVC
from sklearn.metrics import mean_squared_error, accuracy_score, r2_score    # function to calculate mean squared
error

df = pd.read_excel("earnings.xlsx")

print(df)

external_traffic_percentage = df["External Traffic Percentage"].to_numpy()
topics = df["Topic"].to_numpy()
answers = df["Answers"].to_numpy()
views = df["Views"].to_numpy()
earnings = df["Earnings"].to_numpy()

fig, axes = plt.subplots(figsize=(10,10))
axes.scatter(answers, earnings);
axes.set_xlabel("Number of Answers")
axes.set_ylabel("Earnings (¥)")
axes.set_title("Number of answers and earnings scatterplot")
plt.show()

fig, axes = plt.subplots(figsize=(10,10))
axes.scatter(external_traffic_percentage, earnings);
axes.set_xlabel("External Traffic Percentage")
axes.set_ylabel("Earnings (¥)")
axes.set_title("Question external traffic percentage and earnings scatterplot")
plt.show()

fig, axes = plt.subplots(figsize=(10,10))
axes.scatter(views, earnings);
axes.set_xlabel("Number of Views")
axes.set_ylabel("Earnings (¥)")
axes.set_title("Question number of views and earnings scatterplot")
plt.show()

fig, axes = plt.subplots(1, 1, figsize=(20,10))
axes.scatter(topics, earnings);
axes.set_xlabel("Question topics")
axes.set_ylabel("Earnings (¥)")
axes.set_title("Question topics and earnings scatterplot")
plt.show()

# Huber Regression, Ridge Regression and Linear Regression on earnings - views

tr_error = {} # create a dictionary to store training MSE for both models, regression model will be used as keys
val_error = {} # create a dictionary to store validation MSE for both models, regression model will be used as keys

X_views = df["Views"].to_numpy().reshape(-1,1)
y_earnings = earnings
print(X_views.shape)
print(y_earnings.shape)

X_train, X_val, y_train, y_val = train_test_split(X_views, y_earnings, test_size=0.3, random_state=42)

fig, axes = plt.subplots(1, 1, figsize=(10,10))

HuberModel = HuberRegressor(epsilon = 1).fit(X_train, y_train)
y_pred_train_Huber = HuberModel.predict(X_train)
y_pred_val_Huber = HuberModel.predict(X_val)
tr_error["Huber Regression"] = r2_score(y_train, y_pred_train_Huber)
val_error["Huber Regression"] = r2_score(y_val, y_pred_val_Huber)

RidgeModel = Ridge(alpha=1, solver='svd').fit(X_train, y_train)
y_pred_train_Ridge = RidgeModel.predict(X_train)
y_pred_val_Ridge = RidgeModel.predict(X_val)
tr_error["Ridge Regression"] = r2_score(y_train, y_pred_train_Ridge)
val_error["Ridge Regression"] = r2_score(y_val, y_pred_val_Ridge)

print("The training R squared score is:")
print(tr_error)
print("The validation R squared score is:")
print(val_error)
```

```
axes.set_title("Huber and Ridge Regression Model on earnings-views")
axes.set_xlabel("Number of views")
axes.set_ylabel("Earnings (¥)")
axes.scatter(X_train, y_train, color="blue", label = "Training datapoints")
axes.scatter(X_val, y_val, color="red", label = "Validation datapoints")
axes.plot(X_train, y_pred_train_Huber, color="yellow", linewidth=3, label = "Huber Regression")
axes.plot(X_train, y_pred_train_Ridge, color="green", linewidth=3, label = "Ridge Regression")
axes.legend()
plt.show()


# Huber Regression, Ridge Regression and Linear Regression on earnings - answers

tr_error = {} # create a dictionary to store training MSE for both models, regression model will be used as keys
val_error = {} # create a dictionary to store validation MSE for both models, regression model will be used as keys

X_views = df["Answers"].to_numpy().reshape(-1,1)
y_earnings = earnings
print(X_views.shape)
print(y_earnings.shape)

X_train, X_val, y_train, y_val = train_test_split(X_views, y_earnings, test_size=0.3, random_state=42)

fig, axes = plt.subplots(1, 1, figsize=(10,10))

HuberModel = HuberRegressor(epsilon = 1).fit(X_train, y_train)
y_pred_train_Huber = HuberModel.predict(X_train)
y_pred_val_Huber = HuberModel.predict(X_val)
tr_error["Huber Regression"] = r2_score(y_train, y_pred_train_Huber)
val_error["Huber Regression"] = r2_score(y_val, y_pred_val_Huber)

RidgeModel = Ridge(alpha=1, solver='svd').fit(X_train, y_train)
y_pred_train_Ridge = RidgeModel.predict(X_train)
y_pred_val_Ridge = RidgeModel.predict(X_val)
tr_error["Ridge Regression"] = r2_score(y_train, y_pred_train_Ridge)
val_error["Ridge Regression"] = r2_score(y_val, y_pred_val_Ridge)

print("The training R squared score is:")
print(tr_error)
print("The validation R squared score is:")
print(val_error)

axes.set_title("Huber and Ridge Regression Model on earnings-answers")
axes.set_xlabel("Number of answers")
axes.set_ylabel("Earnings (¥)")
axes.scatter(X_train, y_train, color="blue")
axes.scatter(X_train, y_train, color="blue", label = "Training datapoints")
axes.scatter(X_val, y_val, color="red", label = "Validation datapoints")
axes.plot(X_train, y_pred_train_Huber, color="yellow", linewidth=3, label = "Huber Regression")
axes.plot(X_train, y_pred_train_Ridge, color="green", linewidth=3, label = "Ridge Regression")
axes.legend()
plt.show()


copydf = df.copy()
copydf['Topic'] = copydf['Topic'].map({"Social / News": 0, 'Health / Psychology': 1, 'Culture / Politics': 2,
'Entertainment / Hobbies': 3, 'Life / Habit': 4, 'Technology / Internet': 5, 'Business / Career': 6, 'Love / Relationship':
7, 'Education / Science': 8, 'Beauty / Fashion': 9})
X_topics = copydf["Topic"].to_numpy()
y_earnings = earnings.reshape(-1,1)
LogisticModel = LogisticRegression(max_iter = 1000)
SvmModel = SVC(max_iter = 1000)

# Logistic Regression and SVM Regression on topics - earnings

tr_acc = {} # create a dictionary to store training accuracy for both models, regression model will be used as keys
val_acc = {} # create a dictionary to store validation accuracy for both models, regression model will be used as
keys

X_train, X_val, y_train, y_val = train_test_split(y_earnings, X_topics, test_size=0.3, random_state=42)

LogisticModel.fit(X_train, y_train)
y_pred_train_Logistic = LogisticModel.predict(X_train) # predict the labels of training set
acc_train = accuracy_score(y_train, y_pred_train_Logistic) # calculate the training accuracy
tr_acc["Logistic Regression"] = acc_train # add the training accuracy to the dict

y_pred_val_Logistic = LogisticModel.predict(X_val) # predict the labels of validation set
acc_val = accuracy_score(y_val, y_pred_val_Logistic) # calculate the validation accuracy
val_acc["Logistic Regression"] = acc_val # add the validation accuracy to the dict

SvmModel.fit(X_train, y_train)

y_pred_train_Svm = SvmModel.predict(X_train) # predict the labels of training set
acc_train = accuracy_score(y_train, y_pred_train_Svm) # calculate the training accuracy
tr_acc["SVM Regression"] = acc_train # add the training accuracy to the dict

y_pred_val = SvmModel.predict(X_val) # predict the labels of validation set
acc_val = accuracy_score(y_val, y_pred_val) # calculate the validation accuracy
val_acc["SVM Regression"] = acc_val # add the validation accuracy to the dict

print("The training accuracy is:")
print(tr_acc)
print("The validation accuracy is:")
print(val_acc)

# plot results
fig, axes = plt.subplots(1,3,figsize=(15,4))
```

```
x = np.arange(2)
y1 = list(tr_acc.values())
y2 = list(val_acc.values())
width = 0.2

axes[0].bar(x-0.2, y1, width, color='cyan')
axes[0].bar(x, y2, width, color='orange')
axes[0].set_xticks(x)
axes[0].set_xticklabels(["LogisticRegression","SVC"])
axes[0].set_ylabel("Accuracies")
axes[0].legend(["training accuracy", "validation accuracy"])

X_fit = np.linspace(0, 4000, 1)

axes[1].set_xlabel("feature(mintmp)")
axes[1].set_yticks([0,1])
axes[1].set_ylabel('label (class)')
axes[1].set_title("Logistic Regression")
axes[1].scatter(X_train[:,0],y_train,s=80,c="skyblue",label="training datapoints")
print(y_train)
axes[1].scatter(X_val[:,0],y_val,s=30,c="blue",label="validation datapoints")
axes[1].scatter(X_fit, LogisticModel.predict(X_fit.reshape(-1, 1)),color='r',s=5,label='predicted label ($\hat{y}=1$
if $h(x) > 0$)')
axes[1].legend()

axes[2].set_xlabel("feature(mintmp)")
axes[2].set_yticks([0,1])
axes[2].set_ylabel('label (class)')
axes[2].set_title("SVM Regression")
axes[2].scatter(X_train[:,0],y_train,s=100,c="skyblue",label="training datapoints")
axes[2].scatter(X_val[:,0],y_val,s=30,c="blue",label="validation datapoints")
axes[2].scatter(X_fit, SvmModel.predict(X_fit.reshape(-1, 1)),color='r',s=5,label='predicted label ($\hat{y}=1$ if
$h(x) > 0$)')
axes[2].legend()
plt.show()
```