Chaklam Silpasuwanchai, Akraradet Sinsamersuk
AT82.03: Machine Learning

A1: Predicting Car Price

In this assignment, you will solve a problem, i.e., Chaky company makes some car but he has difficulty setting the price for the car. Please make a simple web-based car price prediction system.

Note: You are ENCOURAGED to work with your friends, but DISCOURAGED to blindly copy other's work. Both parties will be given 0.

Note: Comments should be provided sufficiently so we know you understand. Failure to do so can raise suspicion of possible copying/plagiarism.

Note: You will be graded upon (1) documentation, (2) experiment, (3) implementation.

Note: This is a two-weeks assignment, but start early.

Deliverables: The GitHub link containing the jupyter notebook, a README.md of the github, and the folder of your web application called 'app'.

Task 1. Preparing the datasets - Download the `Car Price` dataset from Google classroom. Perform loading, EDA, preprocessing, model selection, $\cdots$, inference. Grade will be given based on the how well you adhere to best practices. There are some important coding considerations:

- For the feature `owner`, map `First owner` to 1, ..., `Test Drive Car` to 5
- For the feature `fuel`, remove all rows with CNG and LPG because CNG and LPG use a different mileage system i.e., km/kg which is different from kmpl for Diesel and Petrol
- For the feature `mileage`, remove "kmpl" and convert the column to numerical type (e.g., float). Hint: use `df.mileage.str.split`
- For the feature `engine`, remove "CC" and convert the column to numerical type (e.g., float)
- Do the same for `max power`
- For the feature `brand`, take only the first word and remove the rest
- Drop the feature `torque`, simply because Chaky's company does not understand well about it
- You will found out that `Test Drive Cars` are ridiculously expensive. Since we do not want to involve this, we will simply delete all samples related to it.
- Since `selling price` is a big number, it can cause your prediction to be very unstable. One trick is to first transform the label using `log transform`, i.e.,

```
        y = np.log(df['selling_price'])

```

- During inference/testing, you have to transform your predicted y backed before comparing with `y test`, i.e.,

```
        pred_y = np.exp(pred_y)

```

Your final data should look like this:

```
df.head()
```

Python

	brand	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats
0	Maruti	2014	450000	145500	Diesel	Individual	Manual	1	23.4	1248	74	5.0
1	Skoda	2014	370000	120000	Diesel	Individual	Manual	2	21.14	1498	103.52	5.0
2	Honda	2006	158000	140000	Petrol	Individual	Manual	3	17.7	1497	78	5.0
3	Hyundai	2010	225000	127000	Diesel	Individual	Manual	1	23.0	1396	90	5.0
4	Maruti	2007	130000	120000	Petrol	Individual	Manual	1	16.1	1298	88.2	5.0

FIGURE 1. Example of cleaned data

Task 2. Report - In the end of the notebook, please write a 2-3 paragraphs summary deeply discussing and analysing the results. Possible points of discussion:

- Which features are important? Which are not? Why?
- Which algorithm performs well? Which does not? Why? (here, you haven't learned about any algorithms yet, but you can search online a bit and start building an intuition)

Task 3. Deployment - Develop a web-based application that contains the model. Here you will be tasked to self-study how to deploy the model into production. Here are some guidelines:

Here you have multiple options. Those who are veteran web developer may prefer their own web app stack which is welcomed. For those who are new to this realm, you may consider a simpler/one-stop solution rather than learning the traditional/flexible approach.

The goal of this task is to expose/deploy our model for public use via the web interface. The main scenario is the following:

1) Users enter the domain on their browser. They land on your page.
2) (optional) Users may need to navigate to a prediction page.
3) Users read the instruction given on the page that instructs them on how the prediction works.
4) Users find the input form, put in the appropriate data, and click submit. Note that when you receive the inputs from users, be reminded to scale your features before inputting into the model. This scaler should be the same one that you use to scale your training features.
5) Note that if users do not have information on certain field, you have to allow users to skip that field. In that case, we recommend you to fill the missing field with imputation technique you have learned in the class.
6) A moment later (depending on your model and hardware performance), the result is returned and printed below the form.

Deploying aside, the app should work on the local environment (your machine) first. I would suggest you use 'Dash' by 'Plotly' https://dash.plotly.com/ as a one-stop solution. Spend time studying the 'Quick Start' tutorial on the site and also 'Dash Fundamental'. They are essential for you to know how 'Dash' works.

The deliverable for the app would be, in GitHub, you have a folder 'app' with '.Dockerfile', 'docker-compose.yaml' files, and 'code' folder.

Bootstrap: I know Dockerizing the app could be difficult for newcomers, you will get confused when searching for stuff online, especially, when you just trust ChatGPT to give you the right answer. So, for those who want to postpone the process of learning "Docker", here is the Dockerized Dash project link. Don't worry, you will eventually need to do this yourself in this shortcoming weeks. You can not escape this.

Good luck :-)