Chaklam Silpasuwanchai, Akraradet Sinsamersuk, Thamakorn Kiattikaikul
AT82.03: Machine Learning

A1: Predicting Car Price

In this assignment, you will solve a real-world problem. Chaky's company manufactures cars but faces challenges in determining appropriate pricing. Your task is to develop a simple web-based car price prediction system.

Important Notes:

- You are **encouraged** to collaborate with your peers, but **copying** work is strictly prohibited. Both the copier and the source will receive a score of 0.
- Your code should include sufficient comments to demonstrate your understanding. Lack of comments may raise suspicions of plagiarism.
- This assignment spans two weeks, but it is highly recommended to start early.
- **Deliverables:** Submit a GitHub repository containing the Jupyter Notebook, a 'README.md' file, and a folder named 'app' that contains your web application.

Task 1. Data Preparation and Modeling Apply the concepts you have learned in the case study to solve this problem. You may follow these steps:

- **Load the Data** - Download the `Car Price` dataset.
- **Data Cleaning and Preprocessing** - Prepare the dataset for training by following these steps:
  - 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 cleaned dataset should resemble the following:

```
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

- **Model Training** - Select at least three appropriate machine learning algorithms and train your models.
- **Testing** - Evaluate your models using suitable performance metrics.
- **Inference** - Save the best-performing model for use in your web application.

Task 2. Analysis and Report At the end of the Jupyter Notebook, write a 2-3 paragraph summary analyzing your results. Consider discussing the following:

- Which features significantly impact predictions, and which do not? Why?
- Which algorithms performed well, and which did not? Why? (Although you have not studied algorithms in depth yet, you can explore online resources to develop an intuition.)

Task 3. Deployment Develop a web-based application integrating your trained model. You will need to self-study how to deploy a machine learning model for public use. Follow these general guidelines:

1) Users visit your web application via a browser.
2) Users navigate to the prediction page (if applicable).
3) Users read the instructions explaining how the prediction system works.
4) Users enter input values and submit the form. Ensure that input features are scaled using the same scaler used during training.
5) If a user omits certain fields, apply an appropriate imputation technique.
6) The system returns and displays the predicted car price.

Sample of webapplication: Your web application should predict car prices based on the selected features.

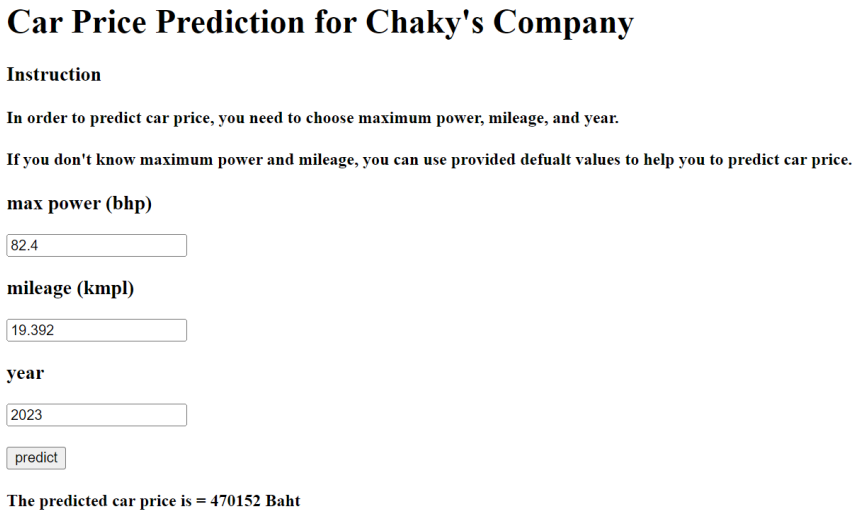

FIGURE 2. Example of web application

Ensure it functions correctly in your local environment before deployment.

Suggested Framework: For an easier implementation, consider using ‘Dash‘ by ‘Plotly‘ https://dash.plotly.com/. Study the ‘Quick Start‘ and ‘Dash Fundamentals‘ tutorials to understand how ‘Dash‘ works.

Deliverables: Your GitHub repository should have structure like this:

```
[[repository name]]/
|— data/
|— code/
|   ├— [[car-price-prediction]].ipynb  # your code
|— app/
|   ├— app.py  # Main script for running the web app
|— .Dockerfile
|— docker-compose.yaml
|— README.md
|— .gitignore
```

FIGURE 3. GitHub file structure

Additional Resources: If you are unfamiliar with Docker, use this pre-configured Dockerized Dash project. However, you are expected to learn Docker within the upcoming weeks.

Good luck! :-)