Rapport d'Erreur

RenneTraffic
28/08/2024
Shaney Carrere

Contexte

Ce projet comprend plusieurs scripts Python (**get_data.py**, **utils.py**, **app.py**) et une interface web avec un fichier **HTML**. Le but du projet est de **prédire le trafic** à l'aide d'un modèle d'intelligence artificielle. Plusieurs erreurs ont été rencontrées et corrigées.

1. Script get_data.py

1.1 Erreur d'Indentation

Description de l'erreur: Le script contenait une erreur d'indentation (**IndentationError**) pour la variable **temp_df.**

Cause: Mauvaise indentation du code.

Solution appliquée:

- Correction de l'indentation pour respecter la structure.

Résultat: Le script s'exécute désormais sans erreurs d'indentation.

1.2 Erreur de Syntaxe

```
def __call__(self):

    res_df = pd.DataFrame({})

    for data_dict in self.data:
        temp_df = self.processing_one_point(data_dict)
        res_df = pd.concat([res_df, temp_df])

        res_df = res_df[res_df.traffic != 'unknown']

        return res_df
```

Description de l'erreur: Une erreur de syntaxe (**SyntaxError**) a été identifiée pour la variable

res_df.

Cause: Syntaxe incorrecte dans l'expression ou la déclaration associée à **res_df.**

Solution appliquée:

- Ajout de **crochets**

Résultat: Le code est maintenant correct et s'exécute sans erreur de syntaxe.

1.3 Problème de la variable traffic_status

Description de l'erreur:

- La variable **traffic_status** était manquante à un endroit du code.
- Une autre occurrence de **traffic_status** nécessitait sa suppression ou modification.

Solution appliquée:

- Suppression de la variable **traffic_status** dans lla liste.
- Renommage de la variable **traffic_status** en **trafficstatus**.

Résultat: Le script utilise désormais la colonne **trafficstatus** de manière cohérente, comme dans les data.

1.4 Renommage des Variables

Description de l'erreur: Les variables latitude et longitude devaient être renommées pour correspondre aux conventions du projet.

Solution appliquée:

- Renommage de **latitude** en **lat**.
- Renommage de **longitude** en **lon**.

Résultat: Les variables de localisation sont désormais cohérentes avec le reste.

2. Script utils.py

2.1 Erreur de Syntaxe

```
def create_figure(data):

    fig_map = px.scatter_mapbox(
            data,
            title="Traffic en temps réel",
            color="traffic",
            lat="lat",
            lon="lon",
            color_discrete_map={'freeFlow':'g
            'heavy':'orange', 'congested':'re
            zoom=10
            height=500,
            mapbox_style="carto-positron"
```

```
            zoom=10
```

```
        zoom=10,
```

Description de l'erreur: Une erreur de syntaxe (**SyntaxError**) a été identifiée pour l'attribution de la variable **zoom=10**.

Cause: Syntaxe incorrecte dans la déclaration.

Solution appliquée:

- Correction de la syntaxe du code pour la déclaration **zoom=10**.

Résultat: Le script s'exécute maintenant correctement sans erreur de syntaxe.

3. Fichier HTML index.html

3.1 Renommage du Fichier HTML

Description de l'erreur: Le fichier **index.html** devait être renommé en **home.html** pour correspondre à la nouvelle convention de nommage.

Solution appliquée:

- Renommage de **index.html** en **home.html.**

Résultat: Le fichier est correctement renommé et référencé dans l'application.

4. Script app.py

4.1 Problème de Type (TypeError) pour la Fonction de Prédiction

TypeError

```
TypeError: prediction_from_model() missing 1 required positional argument: 'hour_to_predict'
```

Description de l'erreur: La fonction **prediction_from_model()** générait une **TypeError** en raison d'un argument manquant **(hour_to_predict).**

Cause: La fonction **prediction_from_model** requiert un argument supplémentaire qui n'était pas fourni lors de son appel.

Solution appliquée:

- Ajustement de l'appel de fonction pour inclure le **deuxième argument** nécessaire :
- **cat_predict = prediction_from_model(model, selected_hour)**

Résultat: La fonction est maintenant appelée correctement **avec les deux arguments** requis, et l'erreur **TypeError** a été résolue.

4.1 Problème de shape (ValueError) pour la Fonction de Prédiction

ValueError

```
ValueError: in user code:

    File "/home/shaney/.local/lib/python3.8/site-packages/keras/src/engine/training.py", line 2341,
in predict_function  *
        return step_function(self, iterator)
    File "/home/shaney/.local/lib/python3.8/site-packages/keras/src/engine/training.py", line 2327,
in step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "/home/shaney/.local/lib/python3.8/site-packages/keras/src/engine/training.py", line 2315,
in run_step  **
        outputs = model.predict_step(data)
    File "/home/shaney/.local/lib/python3.8/site-packages/keras/src/engine/training.py", line 2283,
in predict_step
        return self(x, training=False)
    File "/home/shaney/.local/lib/python3.8/site-packages/keras/src/utils/traceback_utils.py", line
70, in error_handler
        raise e.with_traceback(filtered_tb) from None
    File "/home/shaney/.local/lib/python3.8/site-packages/keras/src/engine/input_spec.py", line
298, in assert_input_compatibility
        raise ValueError(

    ValueError: Input 0 of layer "sequential_31" is incompatible with the layer: expected shape=
(None, 24), found shape=(None, 25)
```

Description de l'erreur: La **shape** fournie dans le code était (none, **25**)

mais le model attend (none, **24**).

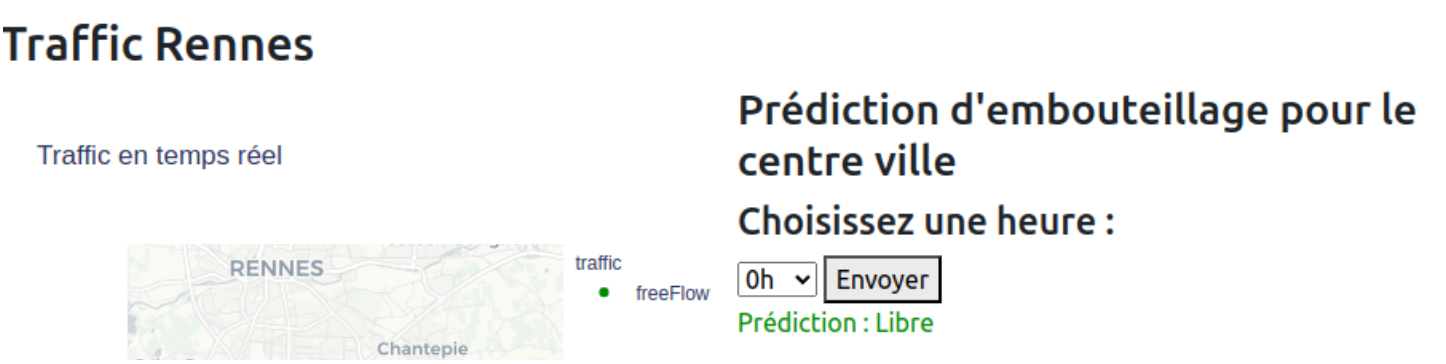
Solution appliquée: remplacé la **shape** par le format attendu par le programme : **(none, 24)**

- Renommage de **index.html** en **home.html.**

```python
def prediction_from_model(model, hour_to_predict):

    input_pred = np.array([0]*24)
    input_pred[int(hour_to_predict)] = 1

    cat_predict = np.argmax(model.predict(np.array([input_pred])))

    return cat_predict
```

Test de l'application :

```
 * Running on http://127.0.0.1:5000
2024-08-27 14:46:31,181 - INFO - [33mPress CTRL+C to quit[0m
2024-08-27 14:46:31,183 - INFO -  * Restarting with stat
2024-08-27 14:46:36,909 - WARNING -  * Debugger is active!
2024-08-27 14:46:36,913 - INFO -  * Debugger PIN: 136-749-603
2024-08-27 14:46:50,739 - INFO - 127.0.0.1 - - [27/Aug/2024 14:46:50] "POST / HTTP/1.1" 200 -
2024-08-27 14:46:52,205 - INFO - 127.0.0.1 - - [27/Aug/2024 14:46:52] "[33mGET /favicon.ico
2024-08-27 14:46:56,384 - INFO - 127.0.0.1 - - [27/Aug/2024 14:46:56] "POST / HTTP/1.1" 200 -
```



5. Mise en Place des Outils de Développement

5.1 Création du fichier requirements.txt et d'un environnement virtuel

Action:

- Création d'un fichier **requirements.txt** listant toutes les **dépendances du projet**.

- Mise en place d'un environnement virtuel (**venv**) pour gérer les dépendances.

5.2 Configuration de error.log avec Logging

Action:

- Mise en place d'un fichier **error.log** pour enregistrer les erreurs et les événements de l'application.
- Ajout du module **logging** dans le fichier **app.py**.

Résultat: Les erreurs sont désormais journalisées dans **error.log .**

Conclusion

Les erreurs identifiées durant le développement ont été corrigées, permettant au projet de fonctionner correctement. Les ajustements apportés ont également amélioré la gestion des dépendances et le suivi des erreurs, rendant l'application plus robuste et plus facile à maintenir.