EVAT Data Science Team Guide and Tutorial

As part of the EVAT Data Science Team, you play a pivotal role in driving insights and building predictive models to enhance electric vehicle (EV) adoption in Australia. This guide serves as a step-by-step resource for your tasks, from data collection to analytics, AI/ML modeling, and integration with the EVAT app.

1. Data Science Workflow

1.1 Steps in Analytics

1. Data Collection:
  - Source: MongoDB for charger data, Research papers, Government websites, EV chargers specific app/website API (Open street map and Open charge map API).
  - Tools: Python libraries (pandas, requests) for accessing and processing data.

2. Data Cleaning:
  - Handle missing values, outliers, and inconsistencies.
  - Tools: pandas, NumPy.

3. Data Transformation:
  - Feature engineering (e.g., time-based features, geospatial features).
  - Normalization/scaling for ML models.
  - Tools: sklearn, pandas.

4. Exploratory Data Analysis (EDA):
  - Analyze trends, distributions, and correlations.
  - Visualize using Power BI dashboards and Python libraries (matplotlib, seaborn).

5. Model Development:
  - AI/ML models for predictions (e.g., demand forecasting, blackspot identification).
  - Start with simple algorithms (linear regression, decision trees) and scale up to advanced models (LSTMs, Transformers).
  - Tools: scikit-learn, TensorFlow, PyTorch.

6. Evaluation and Optimization:
  - Metrics: RMSE, accuracy, F1 score, etc.
  - Techniques: Hyperparameter tuning, cross-validation.

7. Integration:
  - Deploy models via REST APIs for app consumption.

- Tools: Flask/Django, Docker, AWS/GCP.

2. Tools and Tech Stack

2.1 Programming and Development

- Python: Core language for data preprocessing, modeling, and integration.
  - IDEs: Jupyter Notebook, Google Colab, PyCharm.
- MongoDB: Primary database for charger and user data.
  - Access via pymongo.
- GitHub: Version control for collaborative coding.
  - Repository Link: [Team GitHub Repository].
  - Use branches for feature development and submit pull requests for reviews.
- Power BI: Interactive dashboards for data visualization and tracking progress.
  - Example Dashboards: Demand trends, user behavior analysis, environmental impact metrics.

2.2 Modeling and Machine Learning

- scikit-learn: For basic machine learning models.
- TensorFlow/PyTorch: For deep learning models (e.g., LSTMs for demand forecasting).
- XGBoost/LightGBM: For ensemble modeling.
- Google Maps API: For geospatial data and charger location integration.

2.3 Deployment

- Flask/Django: For creating APIs to serve ML models.
- Docker: Containerization for consistent deployment environments.
- AWS/GCP: Hosting backend services and APIs.

3. AI/ML Modeling Basics

3.1 Model Building Steps

1. Define the Problem: Clearly outline the use case (e.g., forecasting demand, identifying blackspots).
2. Data Preparation: Ensure features are clean and relevant to the problem.
3. Model Selection:
   - Use scikit-learn for initial experiments.
   - For sequential data, explore LSTMs or Transformers.
   - For tabular data, consider tree-based models (XGBoost).

4. Training and Validation: Split data into training, validation, and test sets.
5. Hyperparameter Tuning: Use grid search, random search, or Bayesian optimization.
6. Evaluation: Validate using domain-specific metrics (e.g., RMSE for demand, classification accuracy for recommendations).

3.2 Making Models Useful for the App

- API Integration: Convert trained models into RESTful APIs using Flask/Django.
- Real-Time Predictions: Ensure low-latency responses for app features like live demand prediction.
- User-Centric Features:
  - Charger recommendations based on location and preferences.
  - Environmental impact tracking (CO_2 emissions saved).

4. Creating Dashboards in Power BI

4.1 Key Metrics

- Demand Trends: Display forecasts for charging station usage.
- Environmental Impact: Show CO_2 savings through EV usage.
- User Behavior Analysis: Visualize peak usage times, popular locations, etc.

4.2 Steps to Build Dashboards

1. Connect Data: Import cleaned datasets from MongoDB.
2. Design Visuals: Use line charts, bar graphs, and maps for intuitive representations.
3. Add Filters: Enable dynamic filtering by date, location, and other attributes.
4. Share Insights: Publish dashboards to the EVAT Power BI workspace for team-wide access.

5. Using GitHub for Collaboration

5.1 Workflow

1. Clone Repository: Use `git clone` to get the latest codebase.
2. Branching:
   - Create feature branches for tasks (e.g., feature/demand-model).
   - Avoid committing directly to the main branch.
3. Commits: Write clear and concise commit messages (e.g., Added EDA for demand trends).
4. Pull Requests: Submit PRs for code reviews before merging to the main branch.

5.2 Tips for Effective Use

- Sync frequently with `git pull` to avoid conflicts.
- Use `.gitignore` to exclude unnecessary files (e.g., datasets, logs).
- Attend GitHub workshops to upskill if needed.

6. Integration with the App

6.1 Workflow

1. APIs for Predictions:
   - Build Flask APIs to serve model predictions.
   - Deploy APIs on GCP/AWS for scalability.
2. Collaboration with App Team:
   - Work with the backend team to integrate APIs.
   - Share model endpoints and usage documentation.
3. Testing and Feedback:
   - Test app functionality with integrated models.
   - Collect feedback to refine predictions and improve app features.

Closing Note

Your contributions as a data scientist are crucial to the EVAT project's success. By following this guide, you will effectively support the team's mission of advancing EV adoption. Let's drive innovation together!