Auto MPG Prediction

1. Project Summary

The goal of this project is to build and compare multiple regression models to predict miles per gallon (MPG) from the Auto MPG dataset, track all experiments with **MLflow**, select the best-performing model, register it in the MLflow Model Registry, and deploy it as a REST API using **Flask**.

We tested **5 regression models**:

1. Linear Regression
2. Ridge Regression
3. Random Forest Regressor
4. Gradient Boosting Regressor
5. XGBoost Regressor

The best model was **Random Forest (version 3 in the MLFlow Registry)**, which was deployed to production.
An example API call returned a prediction of **34.019999 MPG**.

Step 1 — Data Loading & Preprocessing

Goal: Load the dataset, clean it, and prepare training/test splits.

Key tasks:

1. Load `auto-mpg.csv`, treating `?` as missing values.
2. Check and impute missing **horsepower** values with the mean.
3. Drop rows missing the target **mpg**.
4. Separate features (X) and target (y).
5. Identify numeric and categorical columns.
6. Split into training and test sets.

In [2]:
```
df
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790.0	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52.0	2130.0	24.6	82	2	vw pickup
395	32.0	4	135.0	84.0	2295.0	11.6	82	1	dodge rampage
396	28.0	4	120.0	79.0	2625.0	18.6	82	1	ford ranger
397	31.0	4	119.0	82.0	2720.0	19.4	82	1	chevy s-10

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

In [7]:
```
df['horsepower'].fillna(df['horsepower'].mean(), inplace=True)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'car_name'],
      dtype='object')
```

In [10]:
```python
import numpy as np
```

In [11]:
```python
# Drop rows with missing target
df = df.dropna(subset=['mpg'])

# Separate features and target
X = df.drop(columns=['mpg','car_name'])
y = df['mpg']

# Identify numerical and categorical columns
num_cols = X.select_dtypes(include=np.number).columns.tolist()
cat_cols = X.select_dtypes(exclude=np.number).columns.tolist()

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In [12]:
```python
print(X_train.shape, X_test.shape)
```

```
(318, 7) (80, 7)
```

Step 2 — Model Training & MLflow Tracking

Goal: Train 5 regression models, log metrics and artifacts with MLflow.

Models:

- Linear Regression
- Ridge Regression
- Random Forest
- Gradient Boosting
- XGBoost

To view MLflow UI:

Visit http://4.227.222.56:5000

jupyter Capstone-Code Last Checkpoint: 25 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

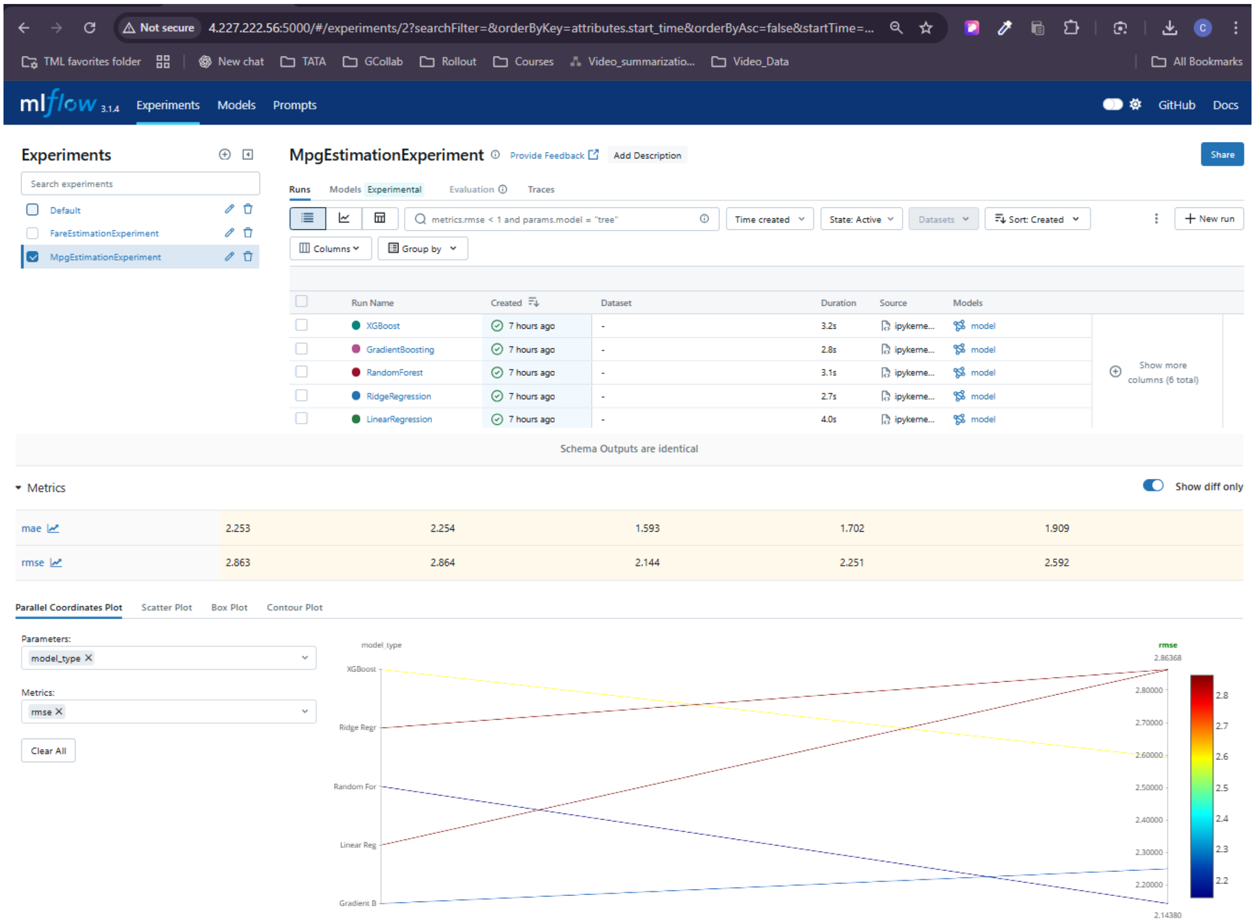
Run | Code

```
mlflow.set_experiment("MpgEstimationExperiment")

with mlflow.start_run(run_name="LinearRegression"):
    model = LinearRegression()
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    rmse = np.sqrt(mean_squared_error(y_test, predictions))
    mae = mean_absolute_error(y_test, predictions)

    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_param("model_type", "Linear Regression")

    signature = infer_signature(X_train, model.predict(X_train))
    mlflow.sklearn.log_model(
        sk_model=model,
        artifact_path="model",
        signature=signature,
        input_example=X_train,
        registered_model_name="MpgEstimator"
    )
```

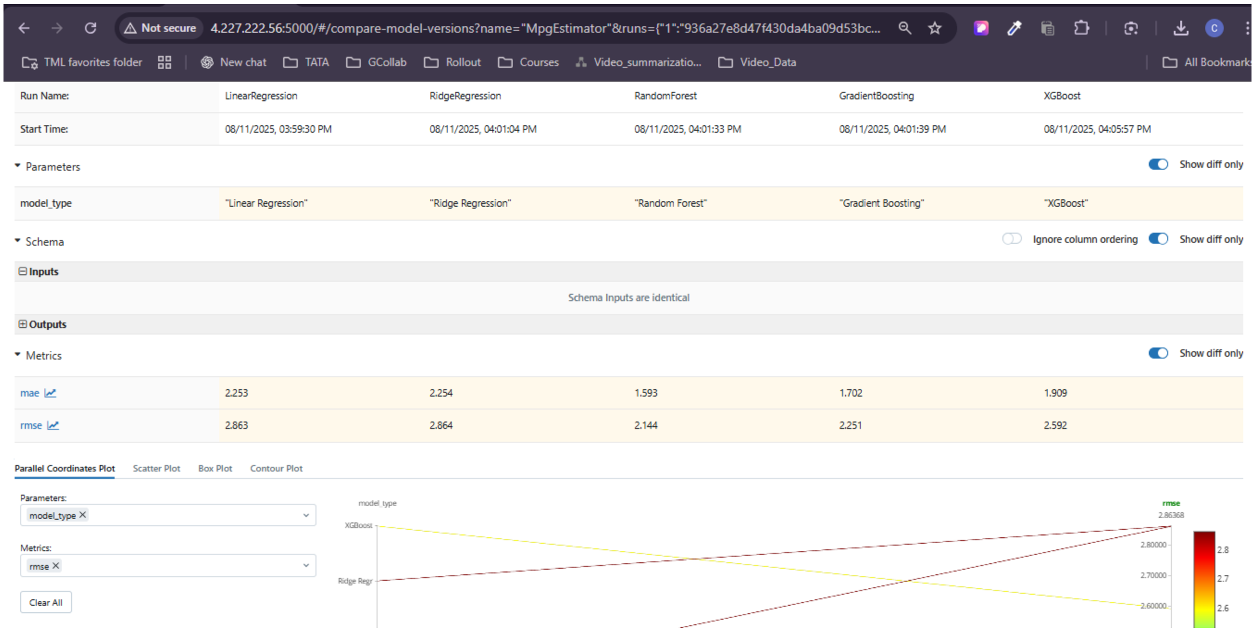
```
2025/08/11 10:29:30 INFO mlflow.tracking.fluent: Experiment with name 'MpgEstimationExperiment' does not exist. Creating a new
experiment.
/home/azureuser/anaconda3/lib/python3.9/site-packages/mlflow/types/utils.py:452: UserWarning: Hint: Inferred schema contains in
teger column(s). Integer columns in Python cannot represent missing values. If your input data contains missing values at infer
ence time, it will be encoded as floats and will cause a schema enforcement error. The best way to avoid this problem is to inf
er the model schema based on a realistic data sample (training dataset) that includes missing values. Alternatively, you can de
clare integer columns as doubles (float64) whenever these columns may have missing values. See `Handling Integers With Missing
Values <https://www.mlflow.org/docs/latest/models.html#handling-integers-with-missing-values>`_ for more details.
  warnings.warn(
2025/08/11 10:29:30 WARNING mlflow.models.model: `artifact_path` is deprecated. Please use `name` instead.
Successfully registered model 'MpgEstimator'.
2025/08/11 10:29:34 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version to finish crea
tion. Model name: MpgEstimator, version 1
```

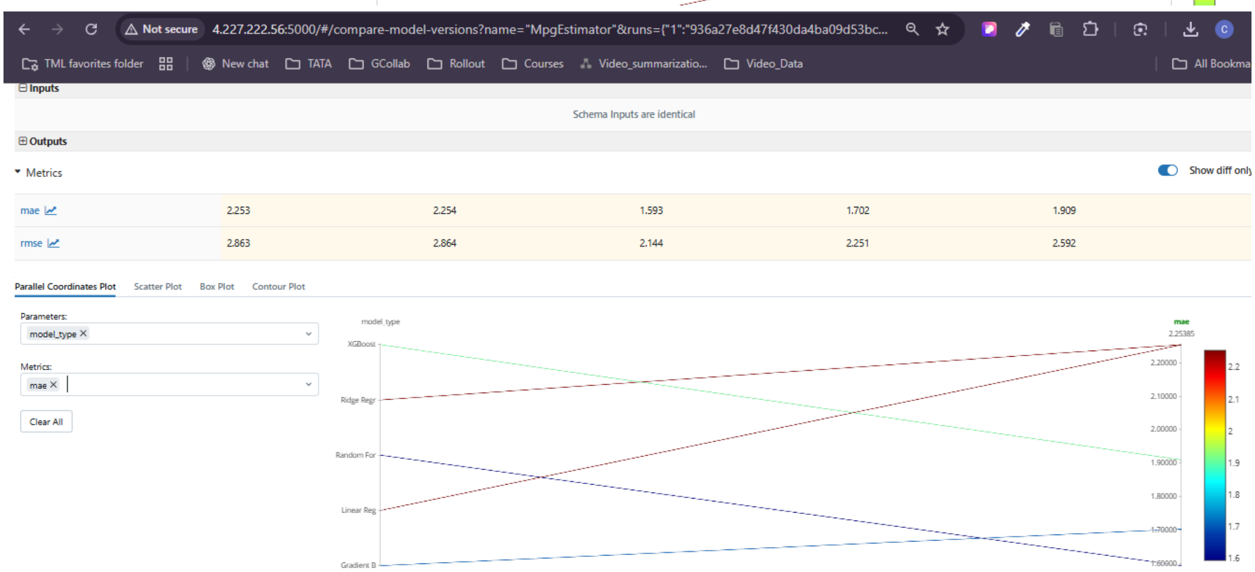
🏃 View run LinearRegression at: http://4.227.222.56:5000/#/experiments/2/runs/936a27e8d47f430da4ba09d53bca7d6f
🧪 View experiment at: http://4.227.222.56:5000/#/experiments/2

```
Created version '1' of model 'MpgEstimator'.
```



Not secure 4.227.222.56:5000/#/experiments/2?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=...
TML favorites folder
New chat
TATA
GCollab
Rollout
Courses
Video_summarizatio...
Video_Data
All Bookmarks
mlflow 3.1.4
Experiments
Models
Prompts
GitHub
Docs
Experiments
Search experiments
Default
FareEstimationExperiment
MpgEstimationExperiment
MpgEstimationExperiment
Provide Feedback
Add Description
Share
Runs
Models Experimental
Evaluation
Traces
metrics.rmse < 1 and params.model = "tree"
Time created
State: Active
Datasets
Sort: Created
New run
Columns
Group by
Run Name
Created
Dataset
Duration
Source
Models
XGBoost
7 hours ago
3.2s
ipykerne...
model
GradientBoosting
7 hours ago
2.8s
ipykerne...
model
RandomForest
7 hours ago
3.1s
ipykerne...
model
RidgeRegression
7 hours ago
2.7s
ipykerne...
model
LinearRegression
7 hours ago
4.0s
ipykerne...
model
Show more columns (6 total)
Schema Outputs are identical
Metrics
Show diff only
mae 2.253 2.254 1.593 1.702 1.909
rmse 2.863 2.864 2.144 2.251 2.592
Parallel Coordinates Plot
Scatter Plot
Box Plot
Contour Plot
Parameters:
model_type
Metrics:
rmse
Clear All
model_type
XGBoost
Ridge Regr
Random For
Linear Reg
Gradient B
rmse
2.86368
2.80000
2.70000
2.60000
2.50000
2.40000
2.30000
2.20000
2.14380
2.8
2.7
2.6
2.5
2.4
2.3
2.2



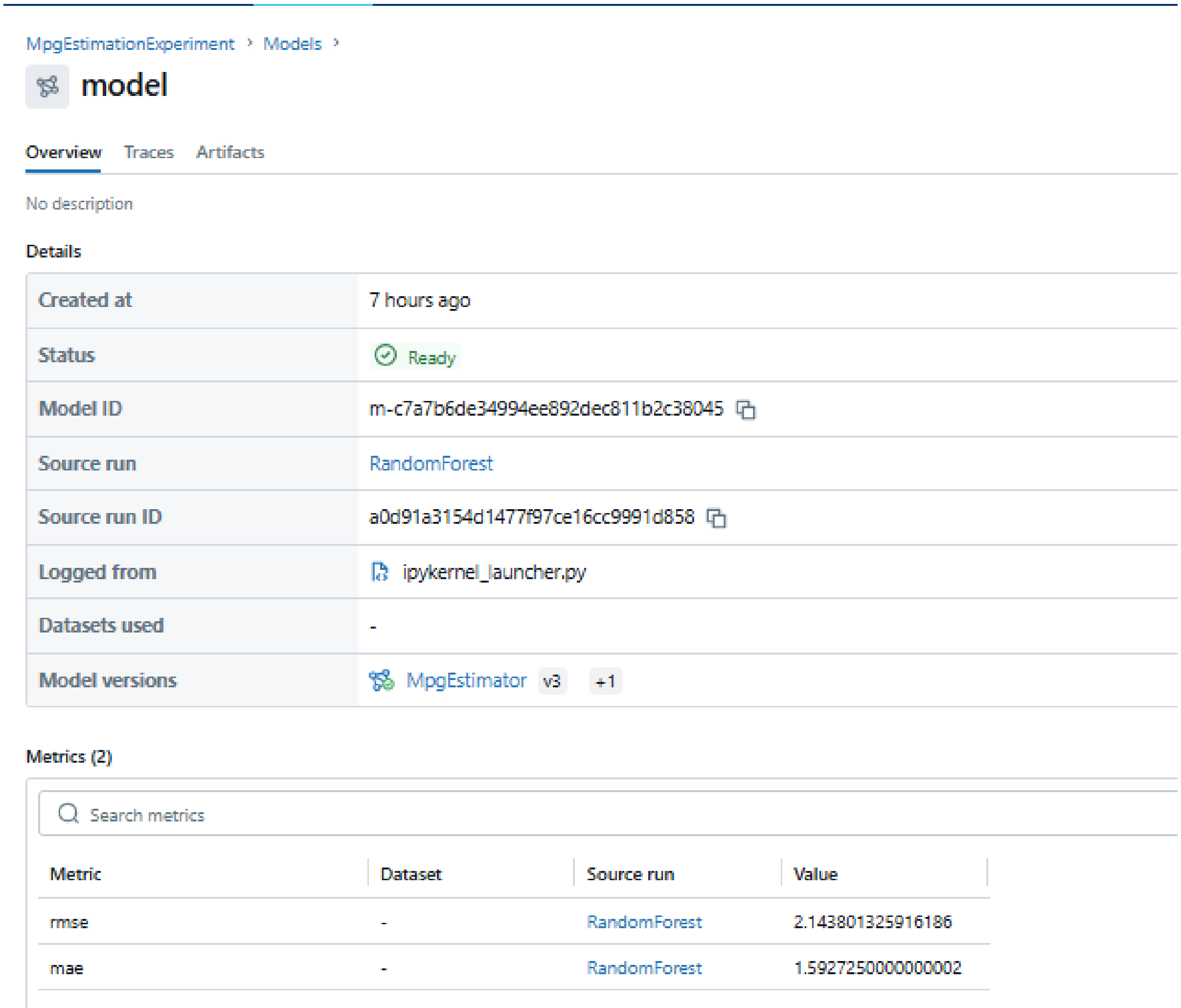


Step 3 — Register the Best Model

After comparing RMSE/MAE in MLflow, Random Forest was chosen the best model as the rmse was the least.

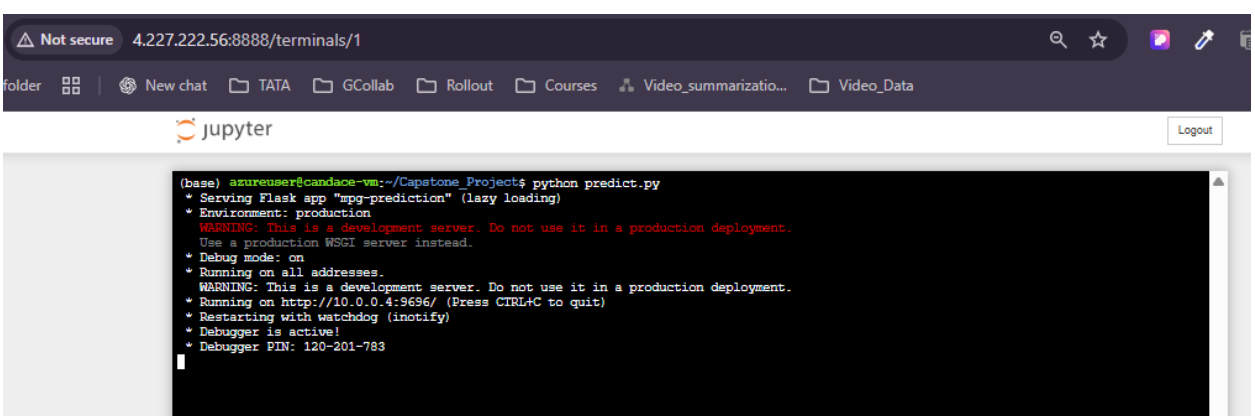
UI Steps:

1. Open Random Forest run in MLflow.
2. Click **Register Model**, name it `RandomForest`.
3. Confirm version number (v3).



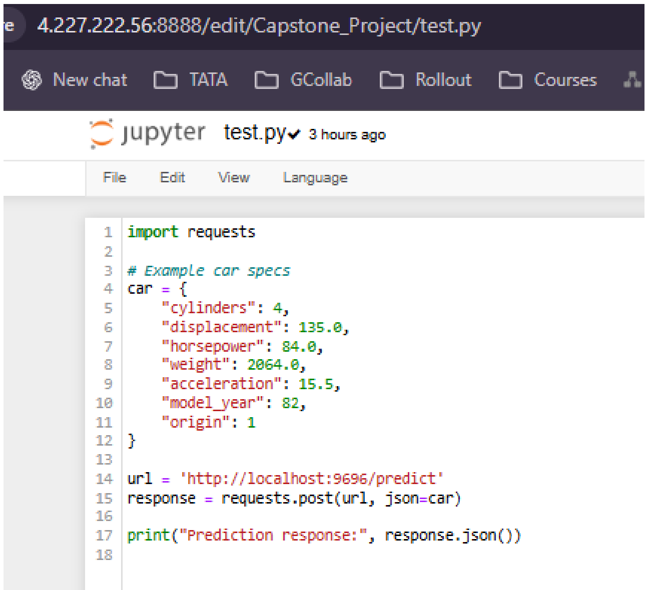
Step 4 — Deployment with Flask

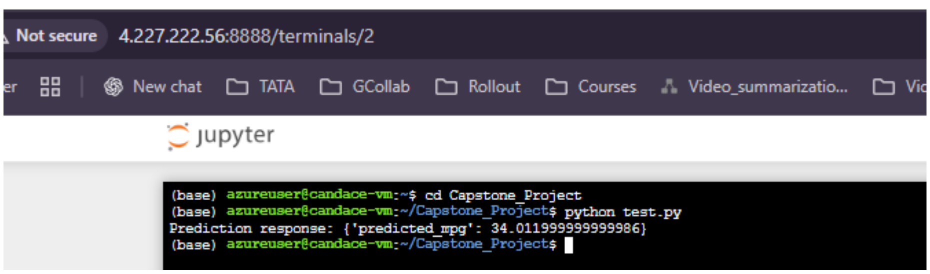
Goal: Load the registered model and expose `/predict` endpoint.



Step 5 — API Testing

test.py:



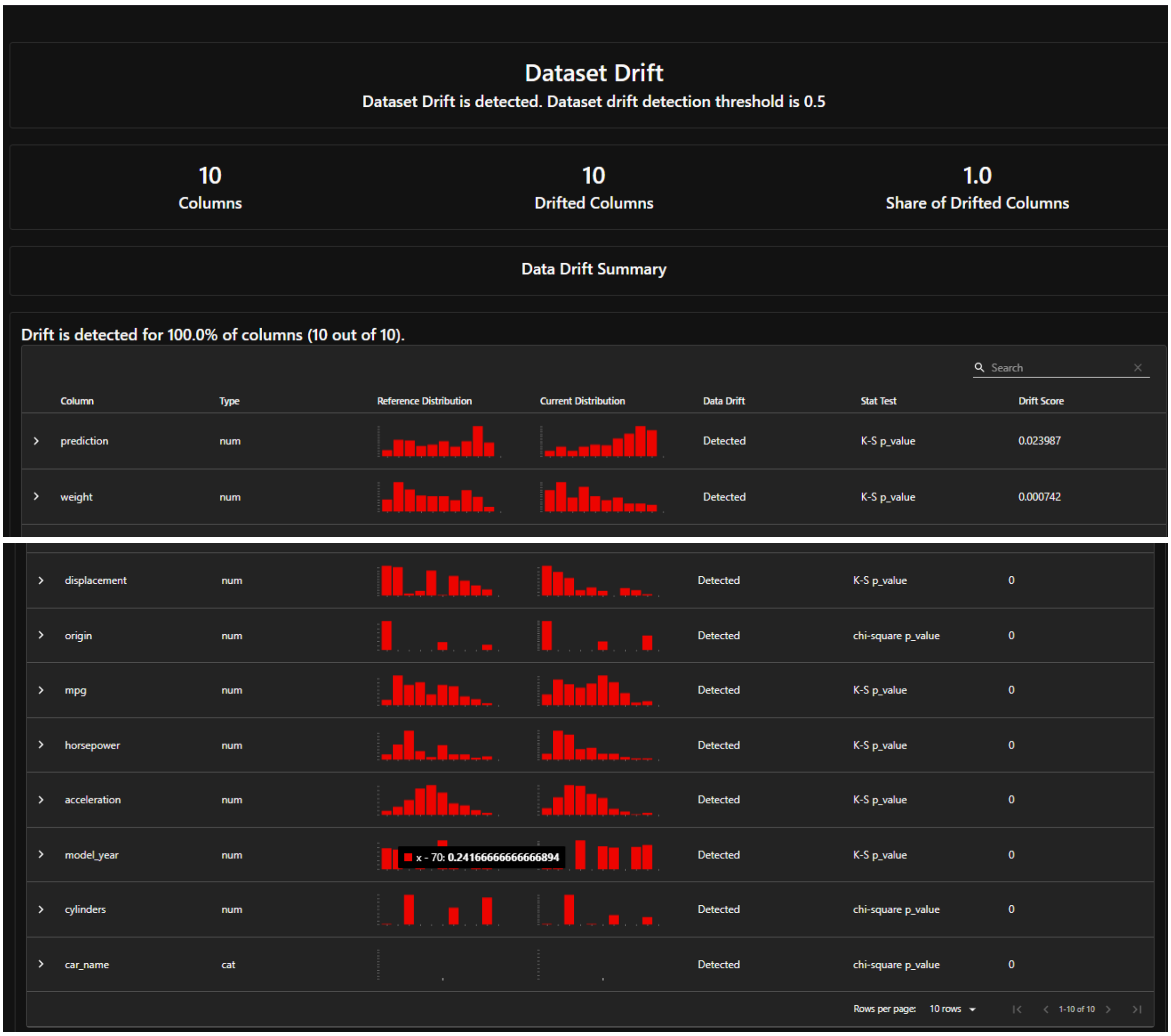


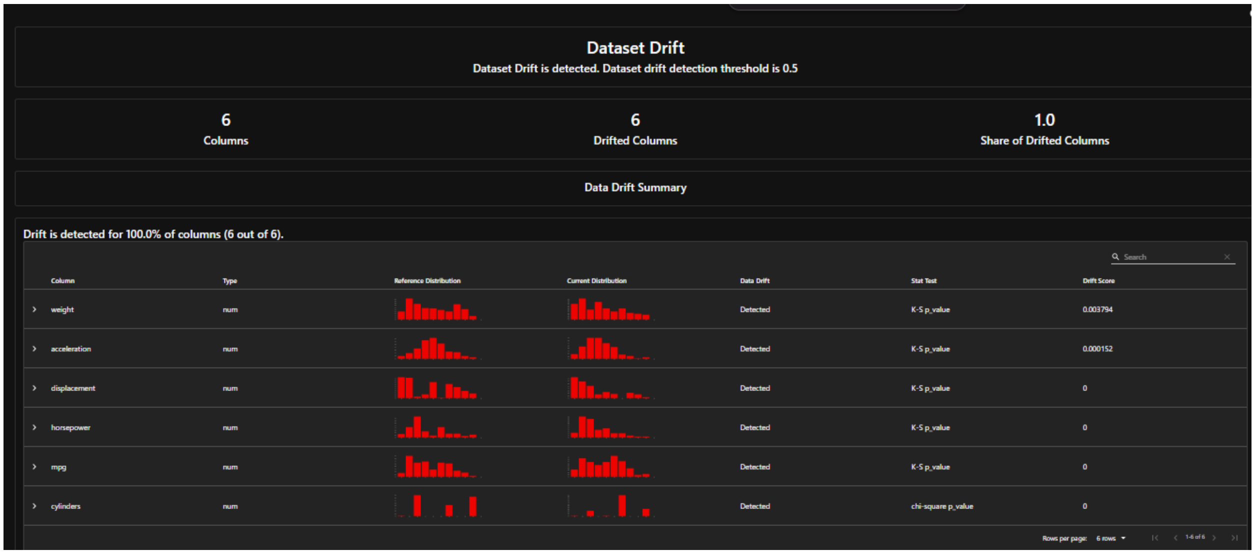
Results Interpretation

After cleaning the Auto MPG dataset and imputing missing horsepower values, five regression models were trained and evaluated. Metrics were logged with MLflow for reproducibility and comparison. Random Forest achieved the best performance and was registered as version 3 in MLflow. The model was successfully deployed via Flask as a REST API, which returned an MPG prediction of 31.863 for the example input.

Evidently Data Drift and Model Performance Report:

This script loads and cleans the Auto MPG dataset, splits it into a baseline (reference) and new (current) dataset, and then deliberately alters the current data to simulate drift in key features like weight, horsepower, and acceleration. It trains a simple linear regression model to predict MPG, generates predictions for both datasets, and uses the Evidently library to create HTML reports on data drift and target drift. Finally, it checks the overall drift score from the report against a set threshold and prints an alert if significant drift is detected.





Evidently Model Performance Report