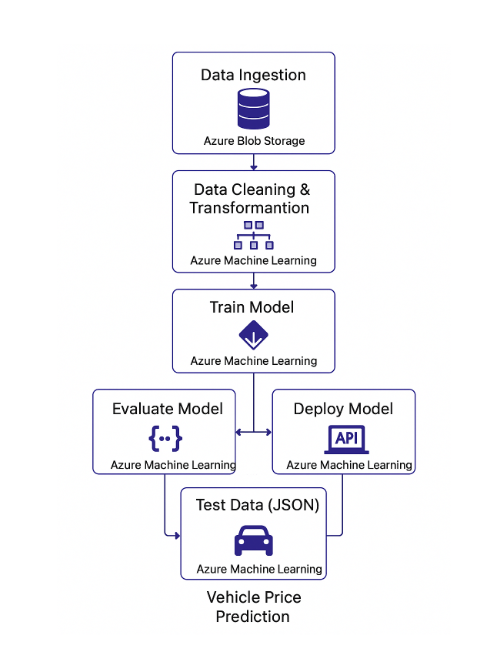
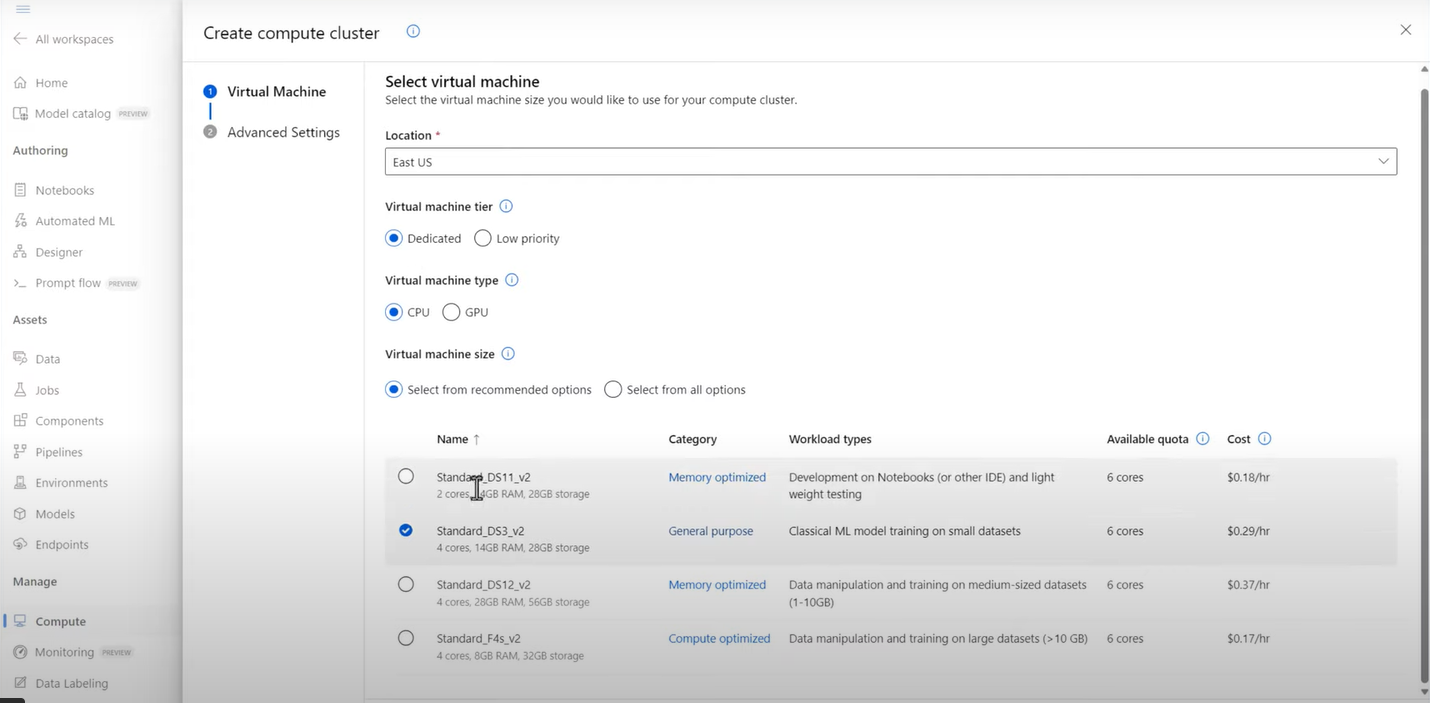
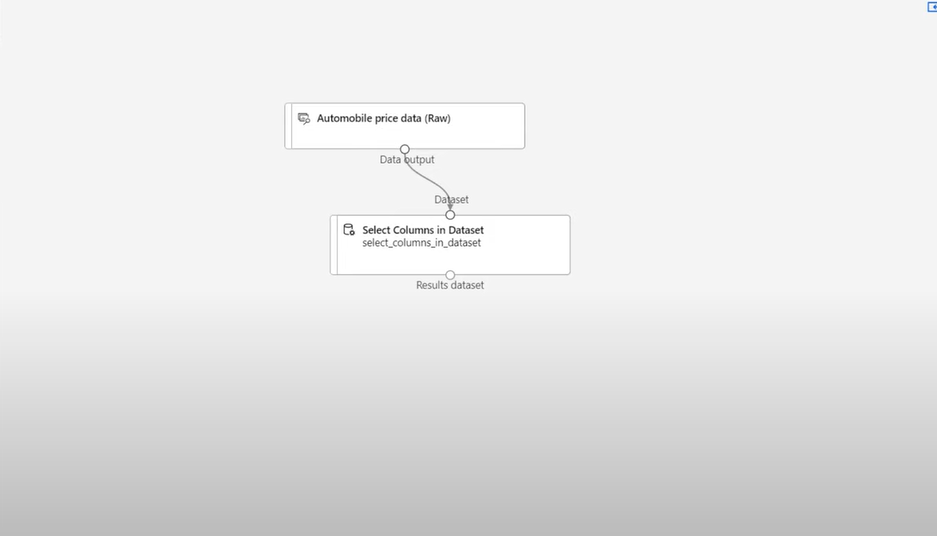
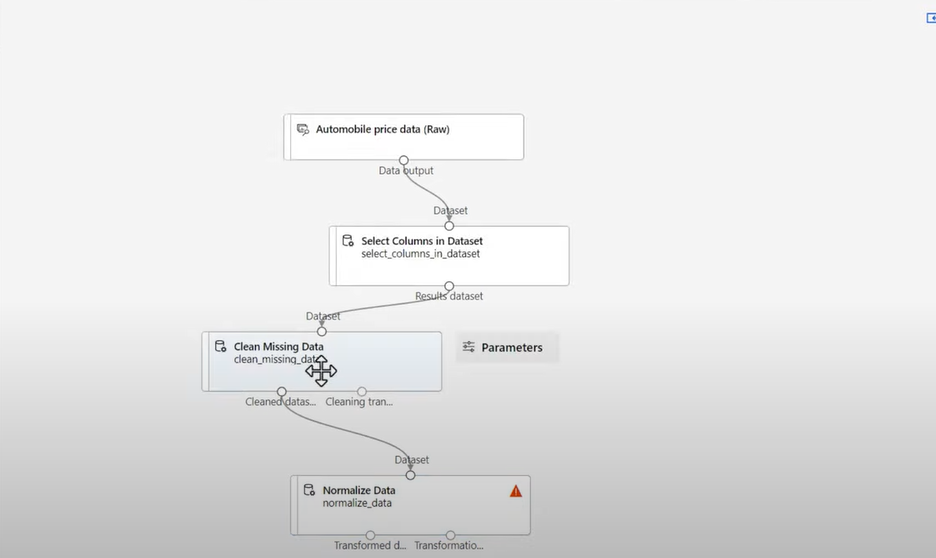
**End-to-End Car Price Prediction Using Azure Machine Learning Pipelines and REST API Deployment.**Azure Services Used

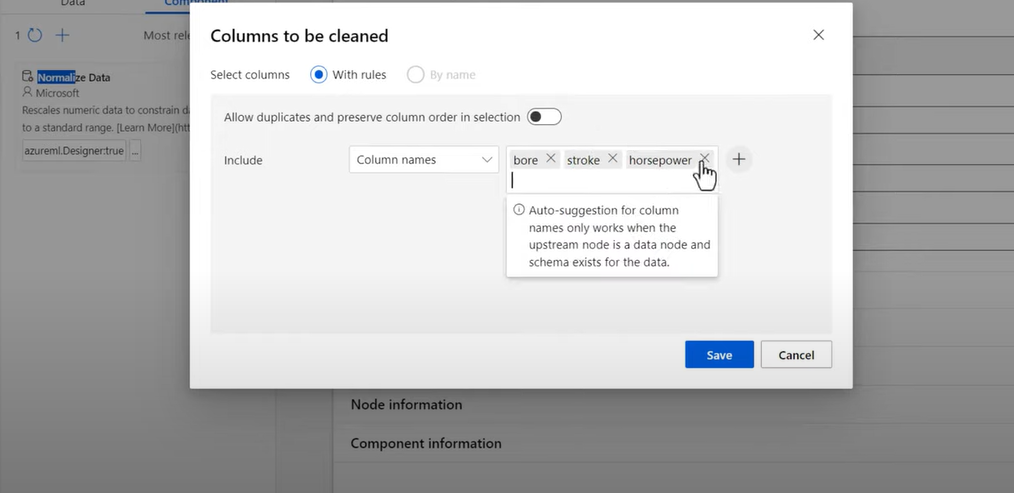
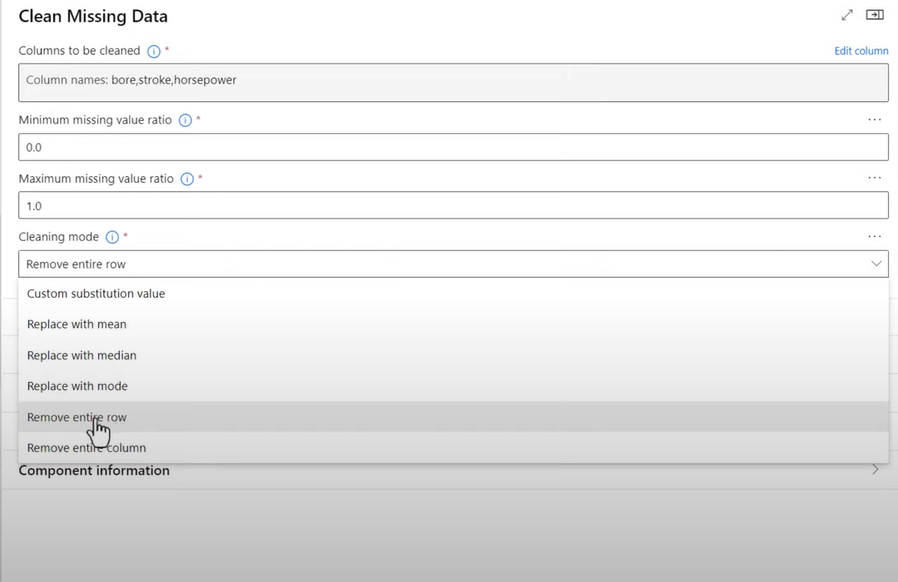
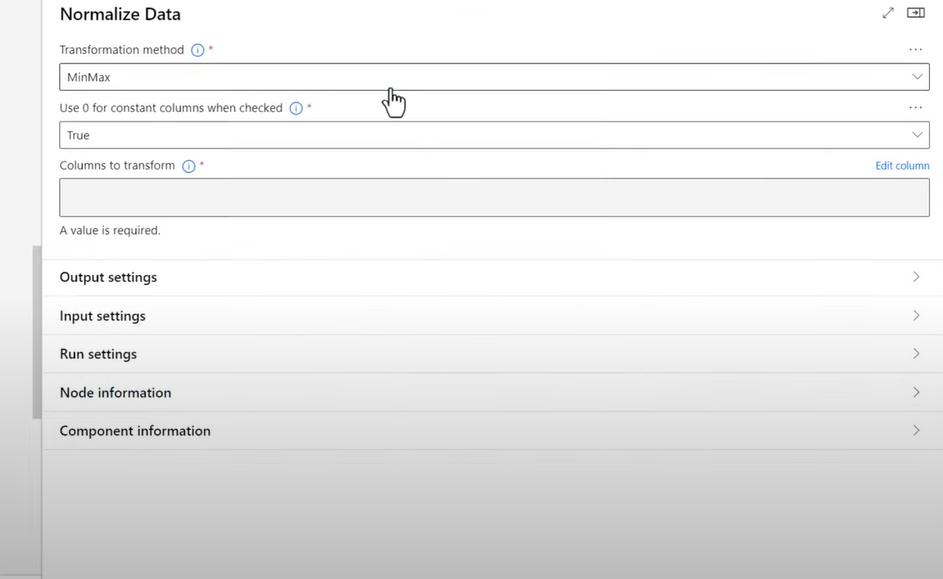
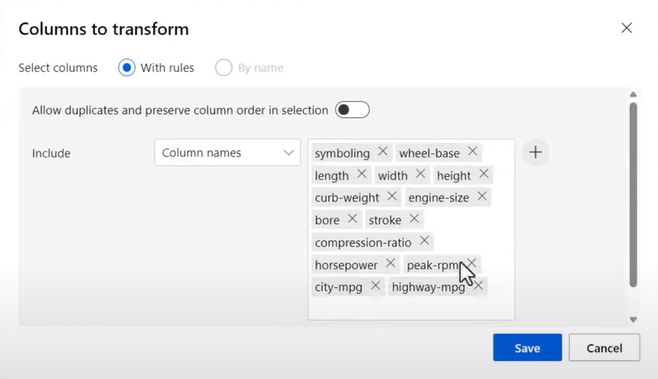
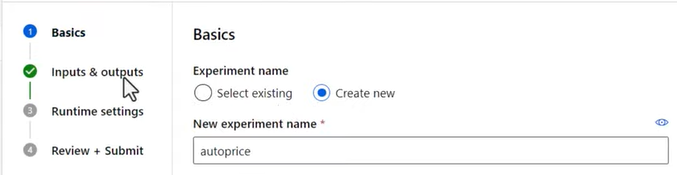
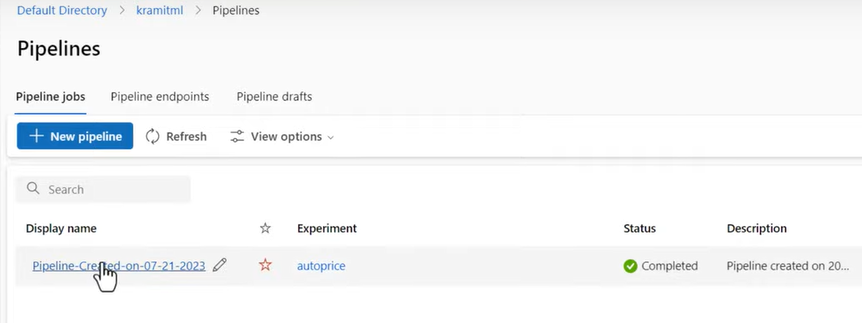
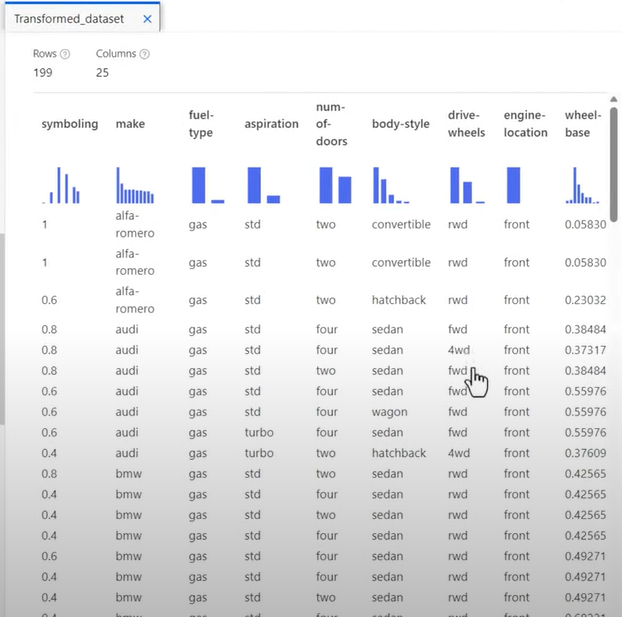
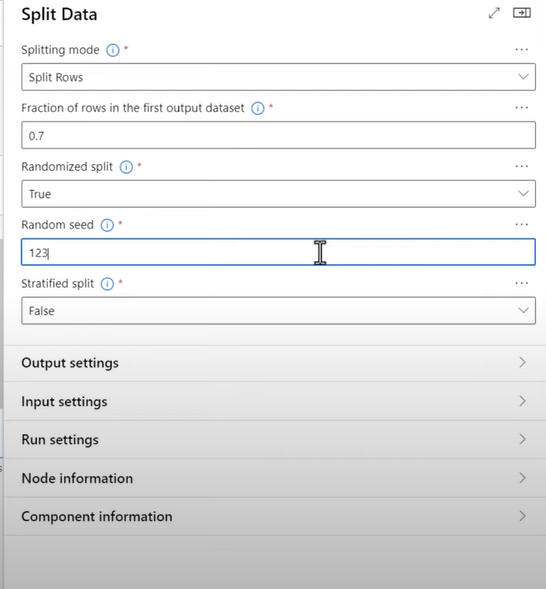
* Azure Machine Learning Studio (Azure ML) – for building, training, and managing ML pipelines and models
* Azure ML Pipelines – for chaining together data preparation, training, evaluation, and deployment steps
* Azure ML Compute Cluster – for scalable, parallel model training
* Azure ML Datastore – for storing datasets in the Azure ML workspace
* Azure ML Dataset – for registering and version-controlling your car price dataset
* Azure ML Managed Online Endpoints – for deploying the trained model as a REST API endpoint
* Azure Key Vault – for managing secrets securely (API keys, access credentials, if needed)
* Azure Storage Account – underlying blob storage for model artifacts, datasets, logs
* Azure Application Insights (optional) – for monitoring deployed endpoint health and metrics
* Visual Studio Code (VS Code) – for local development of Python scripts, pipelines, and scoring scripts
* Swagger UI (optional) – for API testing and documentation

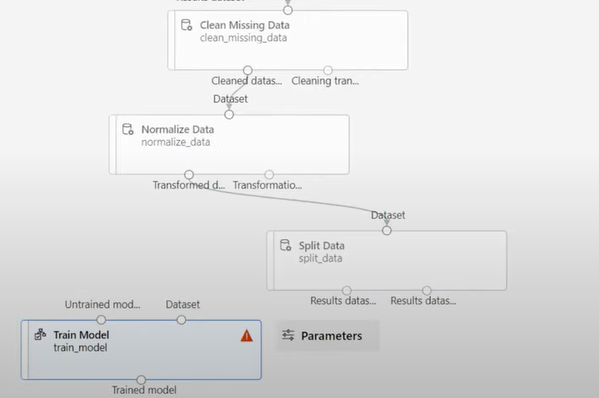
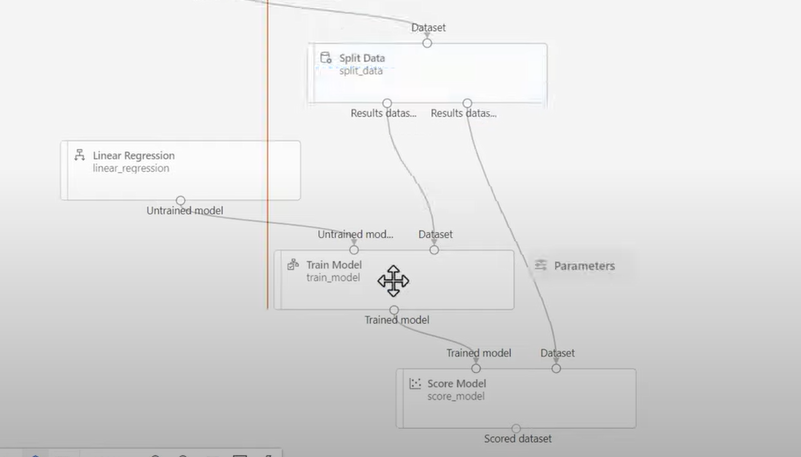
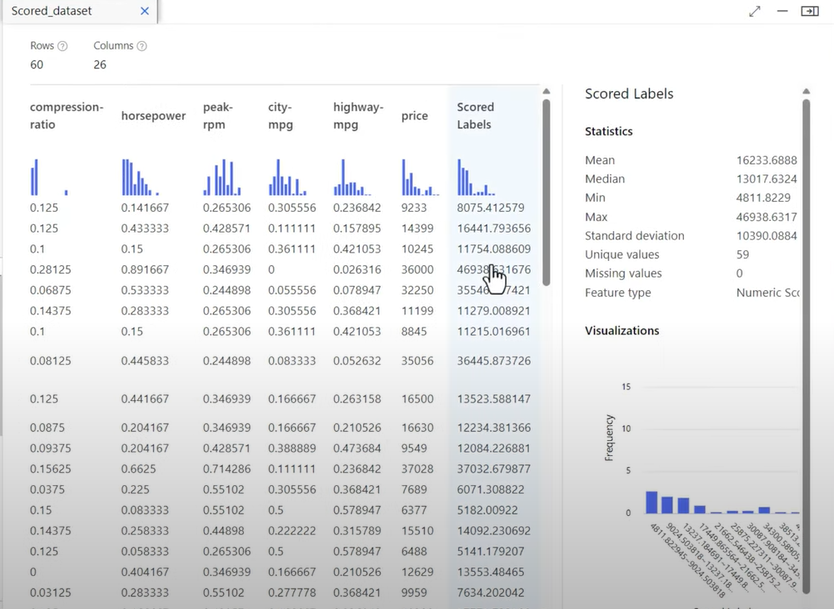
# Machine Learning Concepts Used

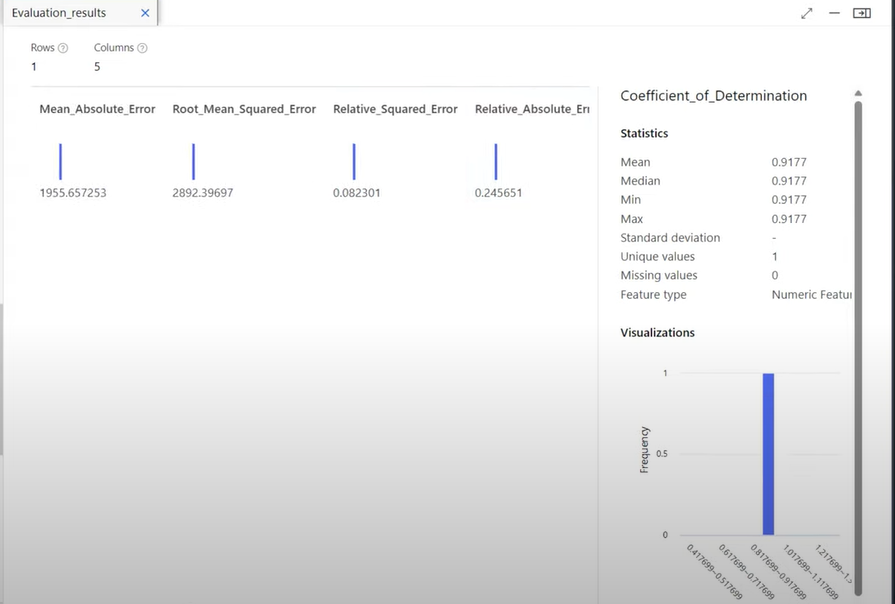
* Feature Engineering:  
  + Categorical encoding (brand, fuel type, color)
  + Numerical feature scaling (Min-Max, Z-Score normalization)
  + Feature selection (removing redundant or low-variance features)
* Data Preprocessing Pipelines:  
  + Modular, reusable cleaning pipelines
  + Separate pipelines for training and production inference
* Model Selection and Training:  
  + Supervised learning – Regression models (Random Forest Regressor, XGBoost Regressor)
  + Train/Test Split for evaluation
  + Cross-validation (optional if HyperDrive was used)
* Model Evaluation:  
  + RMSE (Root Mean Square Error)
  + MAE (Mean Absolute Error)
  + R² Score (Coefficient of Determination)
* Scaling Strategies:  
  + Horizontal scaling using Azure Compute Clusters
  + Parallel pipeline runs for hyperparameter tuning
* Model Deployment:  
  + Deploying model artifacts to a Managed Online Endpoint
  + Using REST API interface with JSON input/output schema
* Monitoring and Governance:  
  + Versioning models and datasets
  + Tracking experiments, metrics, and pipeline runs inside Azure ML workspace.  
      
      
      
      
    

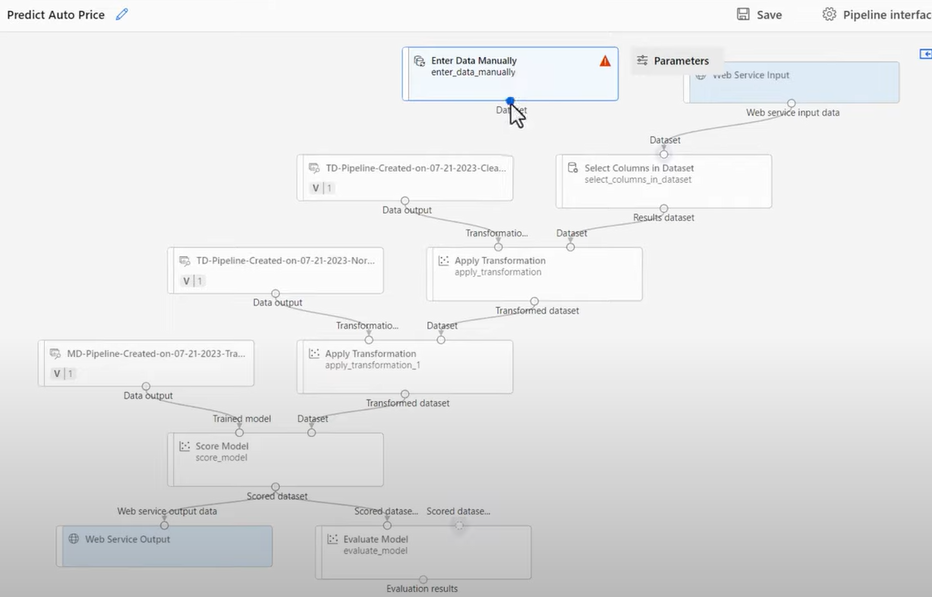
  
  
 After creation of the cluster I am creating a new pipeline in the authoring designer tab in the Azure Machine Learning Studio.  
  
  


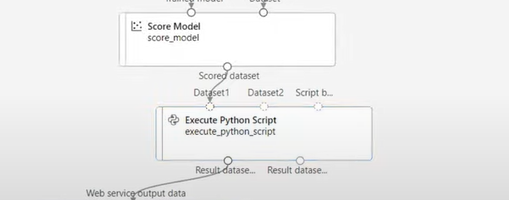
  
  
 Cleaning the data and columns to be cleaned

  
  
  
 Normalize the data by selecting normalized data option in the pipeline.  
  
  
  
  
 Now let's configure and submit this pipeline for the training  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Now let's split the data in the training and testing   
  


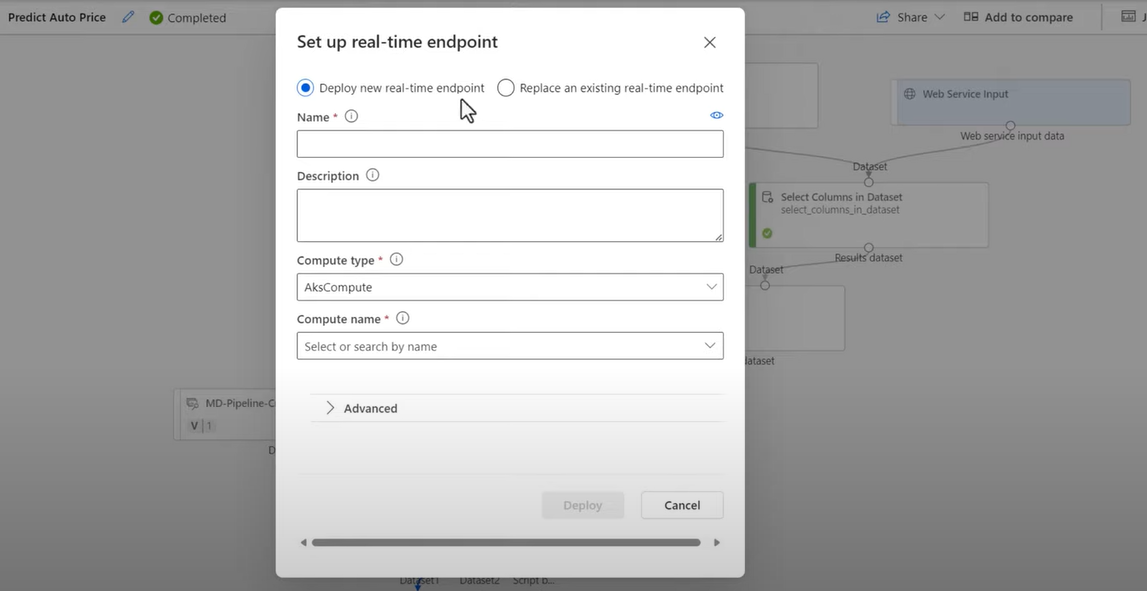
  
  
  
 So after splitting off the data, the data with linear regression as the model then to be used the score model.  
  
  
  
  
  
  
  
 Now let's configure and submit this pipeline  
  


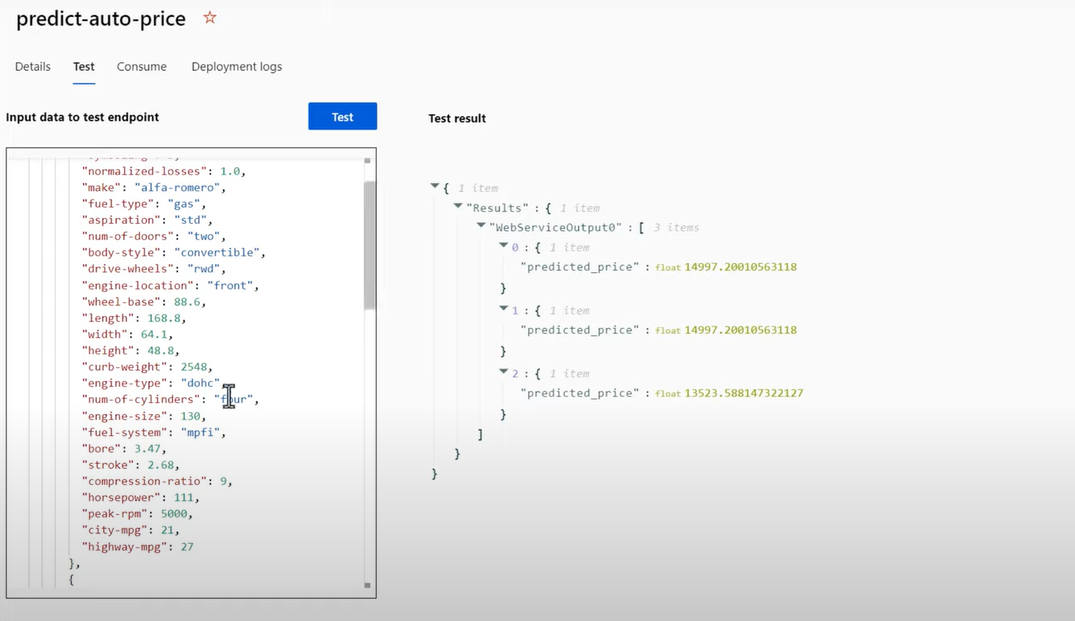
Then we just add evaluate at the end of the pipeline and here are the results.  
  
  
  
  
  
Predict Auto price.  
  
 Now we will use the predict auto pipeline and here we will enter the data manually as a source and connect it to our previous pipeline which has select column and all the rest flow





Click on the deploy model button at the top.



Output of predicted price:  


Project Overview

In this project, I built a complete vehicle price prediction system using Azure Machine Learning (Azure ML) services.  
 The objective was to predict the selling price of a vehicle based on key features like brand, color, horsepower, engine size, fuel type (petrol, diesel, CNG), and other technical and aesthetic parameters.

This project was designed not just as a static model but as a dynamic, production-ready, and API-integrated ML pipeline.

# Architecture and Workflow

1. Dataset Details  
   * Dataset included attributes like brand, model year, mileage, horsepower, transmission type, fuel type, color, engine capacity, and past selling price.
   * Data was uploaded to Azure ML workspace as a registered dataset.
2. Pipeline Components Created  
   * Data Ingestion Step: Read the dataset into Azure ML compute clusters.
   * Data Cleaning and Transformation Step:  
     + Handled missing values.
     + Encoded categorical variables (brand, fuel type, transmission).
     + Applied scaling using both Min-Max normalization and Z-score standardization for numeric features like horsepower and engine size.
   * Data Splitting Step:  
     + Splitted the dataset into training (80%) and testing (20%) sets using a controlled random seed for reproducibility.
   * Model Training Step:  
     + Trained a regression model (Random Forest Regressor and XGBoost models were experimented).
     + Hyperparameters were tuned using Azure ML HyperDrive (if applicable).
   * Model Evaluation Step:  
     + Evaluated performance using RMSE, MAE, and R² metrics.
     + Visualized predicted vs actual prices to assess underfitting/overfitting.
3. Production Pipelines  
   * Created a separate Production Pipeline where all preprocessing steps were reused as a single step inside the final deployed pipeline.
   * The model artifact and the scoring script were integrated into the pipeline.
4. Model Deployment  
   * Deployed the trained model as a real-time scoring endpoint using Azure ML Managed Online Endpoint.
   * Input: JSON format containing car details (brand, horsepower, fuel type, etc.)
   * Output: Predicted vehicle price.
5. Testing the Deployed Model  
   * Used Swagger documentation and custom Python scripts to send test JSON requests.
   * Successfully retrieved price predictions for test car samples via the REST API.

# Challenges Faced

* Complexity in Feature Engineering:  
   Handling a wide variety of categorical and continuous features in a consistent, production-safe way was challenging.  
   I overcame this by building dynamic pipelines with transformers that auto-adapted to schema changes.
* Maintaining Reusability Across Pipelines:  
   It was initially difficult to modularize the data cleaning, transformation, and model training into separate reusable steps.  
   Solved by designing the first cleaning pipeline as an independent reusable component and calling it inside the final production pipeline.
* Scaling Numerical Features Properly:  
   Choosing between Min-Max Scaling and Z-Score Standardization was tricky because different features had wildly different distributions.  
   I created parallel preprocessing pipelines and compared their impact on model evaluation scores before locking the final preprocessing method.
* Model Overfitting on Rare Brands:  
   Some car brands had very few samples, causing the model to overfit.  
   I added feature smoothing, cross-validation, and minimal threshold filtering to avoid noise during model training.
* Deploying with JSON Validation Errors:  
   Early API testing failed because the JSON input schema mismatched the model’s expected feature order and types.  
   Solved by carefully enforcing schema matching during input testing, and setting up schema validation logic inside the scoring script.

# Final Outcome

The final solution was a fully automated, scalable, real-time ML application where:

* Data ingestion, cleaning, transformation, model training, evaluation, and deployment were all done through Azure ML Pipelines.
* The production model was available as an API endpoint that accepted JSON input and returned predicted car prices almost instantly.
* The solution was modular, reusable, highly secure, and cloud-native, showcasing strong ML engineering and MLOps practices.

.