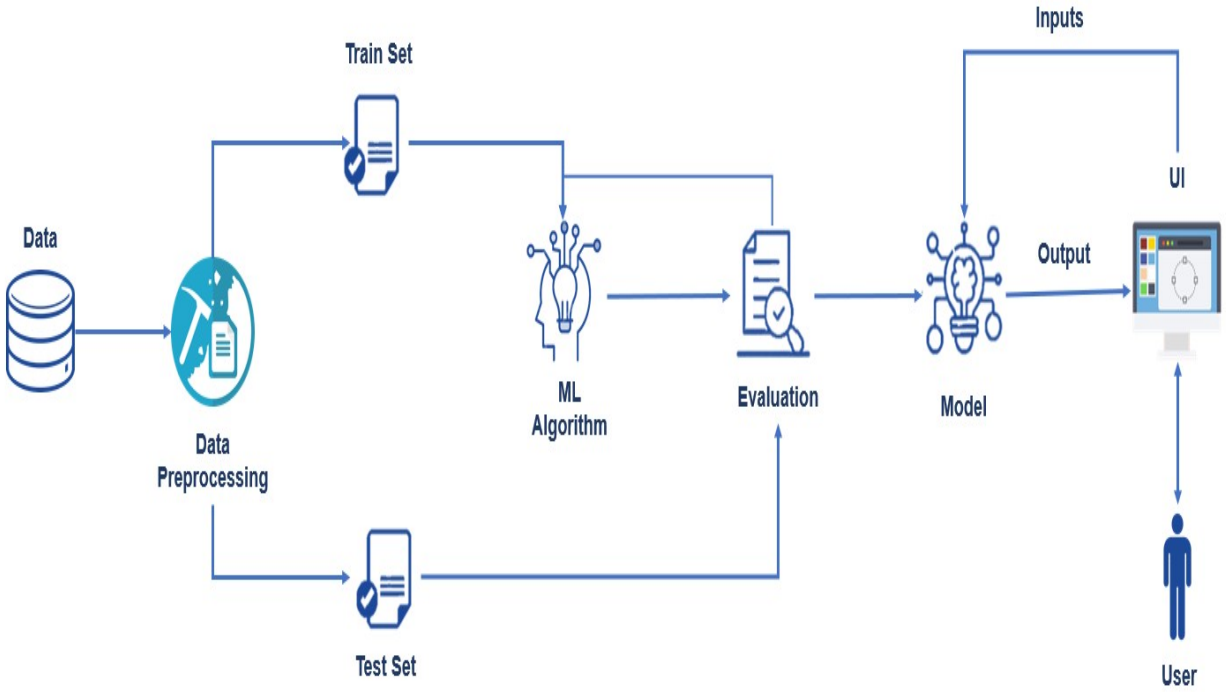
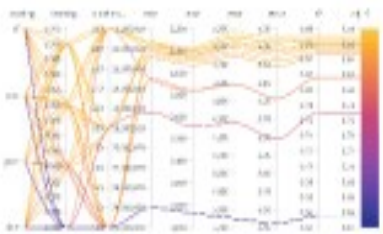
Performance Metrics



Model Performance Testing

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** LGBM Regressor MAE: 1327.55 MSE: 9492244.28 RMSE: 3080.95 RMSLE: 8.03 R2 Score: 0.8668 Adjusted R2 Score: 0.8668	
2.	Tune the Model	**Hyperparameter Tuning** 1) Learning Rate: [0.01, 0.03, 0.05, 0.07] 2) Boosting Type: ['gbdt','dart','goss','rf'] 3) Number of Estimators: [100,200,300] **Validation Method**: Grid Search Cross Validation **Best Parameters:** Learning Rate – 0.07 Boosting Type – 'gbdt' Number of Estimators - 300	 

By applying train test split

Applying Train Test Split

In [35]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

In [74]:
```
from sklearn.linear_model import LinearRegression
```

In [75]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

Creating an OneHotEncoder object to contain all the possible categories

In [39]:
```
ohe=OneHotEncoder()
ohe.fit(X[['name','company','fuel_type']])
```

Out[39]:
```
OneHotEncoder()
```

Creating a column transformer to transform categorical columns

In [52]:
```
column_trans=make_column_transformer((OneHotEncoder(categories=ohe.categories_),['name','company','fuel_type']),
                                     remainder='passthrough')
```

Linear Regression Model

In [54]:
```
lr=LinearRegression()
```

Making a pipeline

In [55]:
```
pipe=make_pipeline(column_trans,lr)
```

Fitting the model

In [59]:
```
pipe.fit(X_train,y_train)
```

Checking R2 Score

Checking R2 Score

In [61]:
```
r2_score(y_test,y_pred)
```

Out[61]:
```
0.7627456237676113
```

Finding the model with a random state of TrainTestSplit where the model was found to give almost 0.92 as r2_score

In [62]:
```
scores=[]
for i in range(1000):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=i)
    lr=LinearRegression()
    pipe=make_pipeline(column_trans,lr)
    pipe.fit(X_train,y_train)
    y_pred=pipe.predict(X_test)
    scores.append(r2_score(y_test,y_pred))
```

In [63]:
```
np.argmax(scores)
```

Out[63]:
```
655
```

In [64]:
```
scores[np.argmax(scores)]
```

Out[64]:
```
0.920088412025344
```

In [65]:
```
pipe.predict(pd.DataFrame(columns=X_test.columns,data=np.array(['Maruti Suzuki Swift','Maruti',2019,100,'Petrol']).reshape(1,5)))
```

Out[65]:
```
array([400707.28215338])
```

The best model is found at a certain random state

In [67]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=np.argmax(scores))
lr=LinearRegression()
pipe=make_pipeline(column_trans,lr)
pipe.fit(X_train,y_train)
y_pred=pipe.predict(X_test)
r2_score(y_test,y_pred)
```

Out[67]:
```
0.920088412025344
```

Metrics

```
model = LGBMRegressor(boosting_type="gbdt",learning_rate=0.07,metric="rmse",n_estimators=300,objective="root_mean_squared_error",random_state=42,reg_sqrt=True)

model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)

find_scores(Y_test, Y_pred, X_train)
```

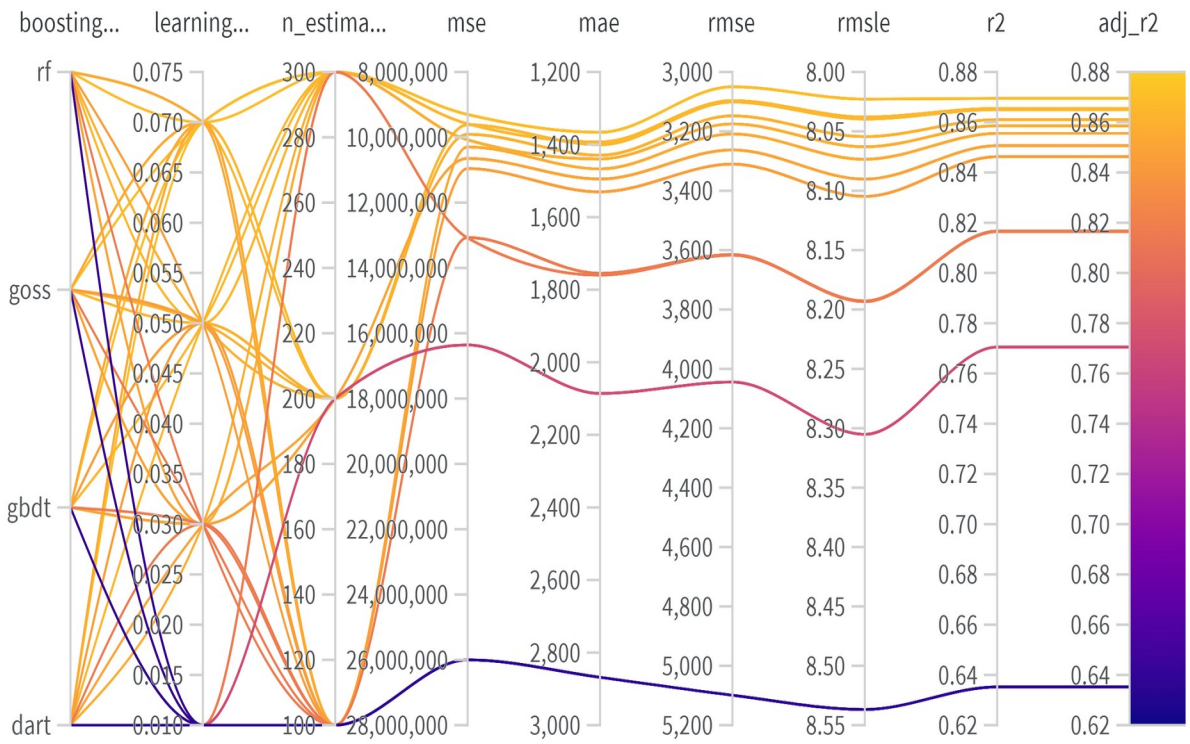
```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was exp
the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

{'mae': 1327.549477341283,
 'mse': 9492244.283543464,
 'rmse': 3080.9406081249859,
 'rmsle': 0.032992815968017,
 'r2': 0.8668348917732229,
 'adj_r2_score': 0.8668269262555739}
```

Tuning the model

```
lgbm_configs = {
    "name":'LGBMRegressor',
    "method": "grid",
    "metric": {
        "name": "adj_r2",
        "goal": "maximize"
    },
    "parameters": {
        "learning_rate": {
            "values": [0.01, 0.03, 0.05, 0.07]
        },
        "objective": {
            "values": ['root_mean_squared_error']
        },
        "boosting_type": {
            "values": ['gbdt','dart','goss','rf']
        },
        "reg_sqrt": {
            "values": [True]
        },
        "metric": {
            "values": ['rmse']
        },
        "n_estimators": {
            "values": [100,200,300]
        },
        "random_state": {
            "values": [42]
        }
    }
}
```

LGBM Regressor



Importing model on IBM :