Training Regression Models on “FirstSession_LTVNumeric” and “FirstDay_LTVNumericTime” Using Time Series and Train-Test Splitting

Introduction

- This specific task covers building of regression models to predict Lifetime Value (LTV) based on first session and first-day user interaction data.

- The goal is to train three different regression models (**Random Forest Regressor**, **Decision Tree Regressor** and **LGBM Regressor**) using time series splitting and train-test splitting techniques to predict LTV based on user data from their first session and first day.

Data Overview

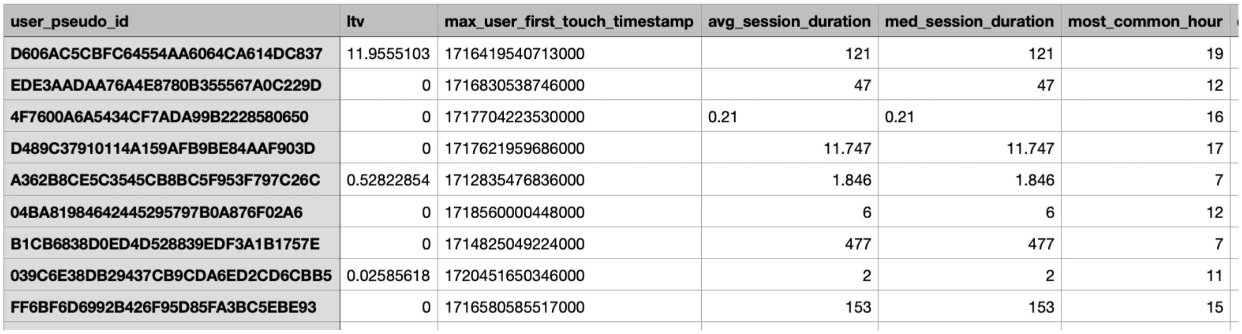
Both data includes various columns such as: avg_session_duration, most_common_hour, user_engagement, transfer_money which would lead prediction of life time value of different users

1- FirstSession_LTVNumeric

- 18583 Rows
- 233 Columns

2- FirstDay_LTVNumeric

- 18718 Rows
- 452 Columns

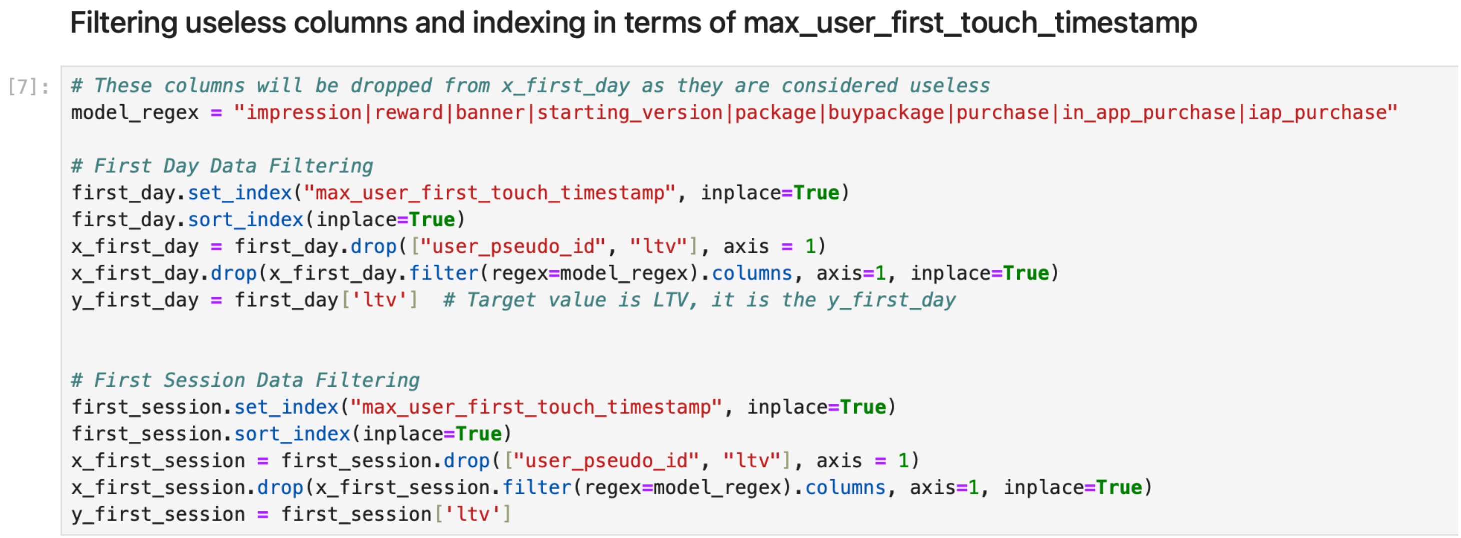


user_pseudo_id	ltv	max_user_first_touch_timestamp	avg_session_duration	med_session_duration	most_common_hour
D606AC5CBFC64554AA6064CA614DC837	11.9555103	1716419540713000	121	121	19
EDE3AADAA76A4E8780B355567A0C229D	0	1716830538746000	47	47	12
4F7600A6A5434CF7ADA99B2228580650	0	1717704223530000	0.21	0.21	16
D489C37910114A159AFB9BE84AAF903D	0	1717621959686000	11.747	11.747	17
A362B8CE5C3545CB8BC5F953F797C26C	0.52822854	1712835476836000	1.846	1.846	7
04BA81984642445295797B0A876F02A6	0	1718560000448000	6	6	12
B1CB6838D0ED4D528839EDF3A1B1757E	0	1714825049224000	477	477	7
039C6E38DB29437CB9CDA6ED2CD6CBB5	0.02585618	1720451650346000	2	2	11
FF6BF6D6992B426F95D85FA3BC5EBE93	0	1716580585517000	153	153	15

Data Import and Preprocessing

The dataset contains multiple features, with some irrelevant ones (like impressions, rewards, banners, etc.) being filtered out using regular expressions.

- **Data Filtering:** Unnecessary columns are removed based on the regex pattern for both datasets.
- **Target Variable:** The target variable (y_first_day and y_first_session) for both datasets is the ltv column which represents Lifetime Value for the users.



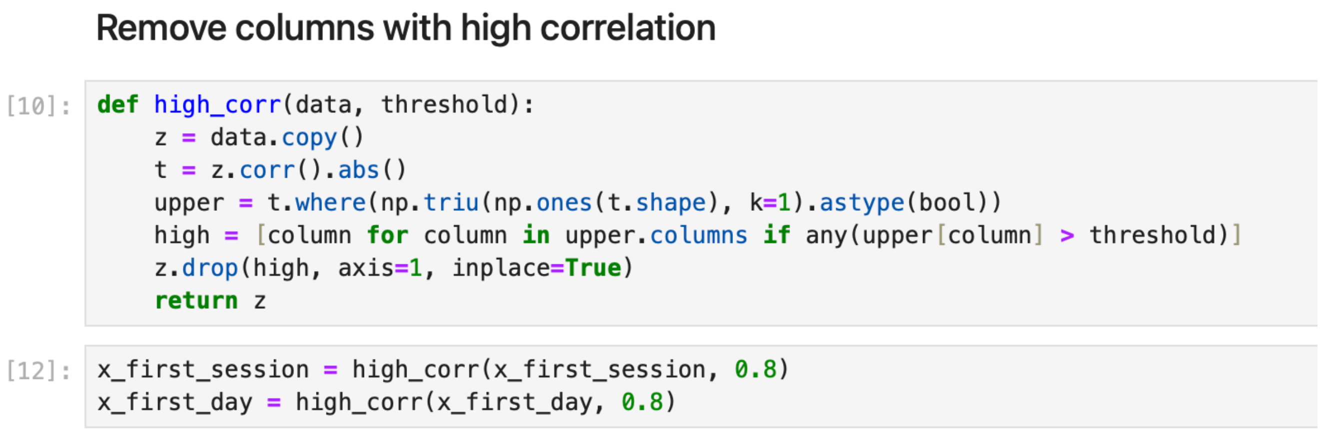
Filtering useless columns and indexing in terms of max_user_first_touch_timestamp

```
[7]: # These columns will be dropped from x_first_day as they are considered useless
model_regex = "impression|reward|banner|starting_version|package|buypackage|purchase|in_app_purchase|iap_purchase"

# First Day Data Filtering
first_day.set_index("max_user_first_touch_timestamp", inplace=True)
first_day.sort_index(inplace=True)
x_first_day = first_day.drop(["user_pseudo_id", "ltv"], axis = 1)
x_first_day.drop(x_first_day.filter(regex=model_regex).columns, axis=1, inplace=True)
y_first_day = first_day['ltv']  # Target value is LTV, it is the y_first_day


# First Session Data Filtering
first_session.set_index("max_user_first_touch_timestamp", inplace=True)
first_session.sort_index(inplace=True)
x_first_session = first_session.drop(["user_pseudo_id", "ltv"], axis = 1)
x_first_session.drop(x_first_session.filter(regex=model_regex).columns, axis=1, inplace=True)
y_first_session = first_session['ltv']
```

- **High Correlation Removal:** High correlation removal helps improve the model's performance and stability by eliminating features that provide redundant information. Therefore the risk of multicollinearity is reduced. In this case, the features with a correlation above 0.8 is removed



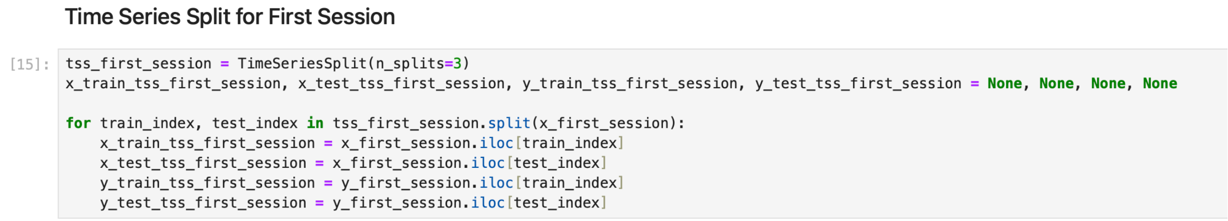
Remove columns with high correlation

```
[10]: def high_corr(data, threshold):
          z = data.copy()
          t = z.corr().abs()
          upper = t.where(np.triu(np.ones(t.shape), k=1).astype(bool))
          high = [column for column in upper.columns if any(upper[column] > threshold)]
          z.drop(high, axis=1, inplace=True)
          return z
```

```
[12]: x_first_session = high_corr(x_first_session, 0.8)
      x_first_day = high_corr(x_first_day, 0.8)
```

Time Series Splitting

For both datasets (first day and first session), the data is split into training and testing sets using TimeSeriesSplit(n_splits=3). This method ensures that the time-dependent nature of the data is maintained.

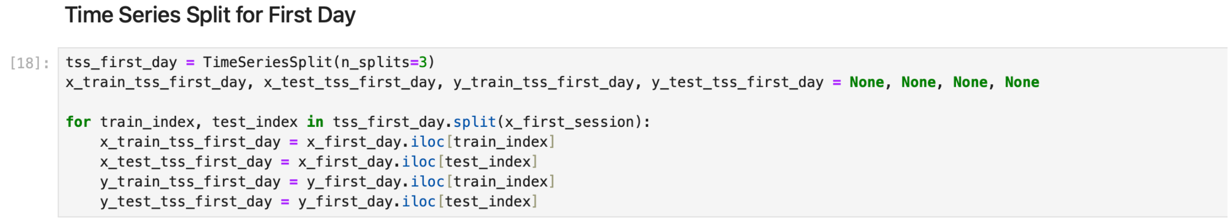


Time Series Split for First Session

[15]:

```
tss_first_session = TimeSeriesSplit(n_splits=3)
x_train_tss_first_session, x_test_tss_first_session, y_train_tss_first_session, y_test_tss_first_session = None, None, None, None

for train_index, test_index in tss_first_session.split(x_first_session):
    x_train_tss_first_session = x_first_session.iloc[train_index]
    x_test_tss_first_session = x_first_session.iloc[test_index]
    y_train_tss_first_session = y_first_session.iloc[train_index]
    y_test_tss_first_session = y_first_session.iloc[test_index]
```



Time Series Split for First Day

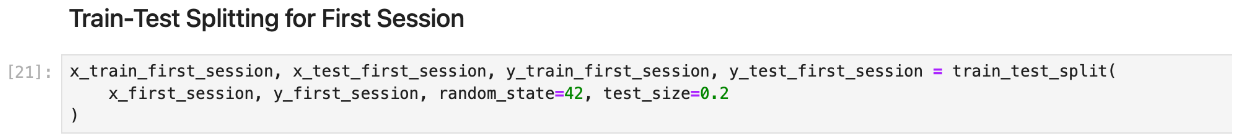
[18]:

```
tss_first_day = TimeSeriesSplit(n_splits=3)
x_train_tss_first_day, x_test_tss_first_day, y_train_tss_first_day, y_test_tss_first_day = None, None, None, None

for train_index, test_index in tss_first_day.split(x_first_session):
    x_train_tss_first_day = x_first_day.iloc[train_index]
    x_test_tss_first_day = x_first_day.iloc[test_index]
    y_train_tss_first_day = y_first_day.iloc[train_index]
    y_test_tss_first_day = y_first_day.iloc[test_index]
```

Train-Test Splitting

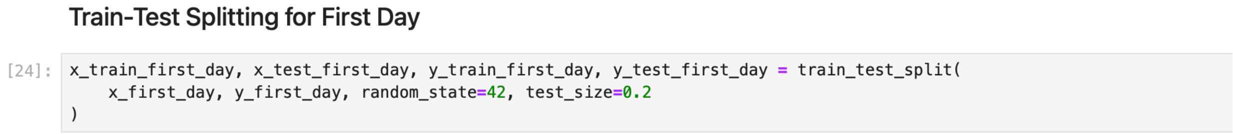
In parallel with time series splitting, the data is also split using **Train-Test Split** (test size = 0.2). This method randomly partitions the data, assuming no time dependencies. This divides the data into two different sections where 80% is named as Train Data and the 20% is the Test Data.



Train-Test Splitting for First Session

[21]:

```
x_train_first_session, x_test_first_session, y_train_first_session, y_test_first_session = train_test_split(
    x_first_session, y_first_session, random_state=42, test_size=0.2
)
```



Train-Test Splitting for First Day

[24]:

```
x_train_first_day, x_test_first_day, y_train_first_day, y_test_first_day = train_test_split(
    x_first_day, y_first_day, random_state=42, test_size=0.2
)
```

Training Regression Models and Evaluation

Following models are used to predict LTV values:

1- Random Forest Regressor
2- Decision Tree Regressor
3- LGBM Regressor

These models are trained and evaluated using both **Train-Test Split** and **Time Series Split**. Key performance metrics such as **R2 Score**, **Mean Squared Error (MSE)**, and **Mean Absolute Error (MAE)** are calculated and displayed.

- **R2 Score**: Indicates the goodness-of-fit.
- **Mean Squared Error**: Measures the average squared difference between actual and predicted values.
- **Mean Absolute Error**: Indicates the average absolute difference between actual and predicted values.

The regression models' results are shown in the table below:

Test - Train Splitting

Regression Models	First Day			First Session		
	R2 Score	Mean Square	Mean Absolute	R2 Score	Mean Square	Mean Absolute
Random Forest Regressor	0,4599	7,7116	0,8835	0,3065	5,0152	0,9771
Decision Tree Regressor	0,4402	7,9923	0,9422	0,2462	5,4507	1,0046
LGBM Regressor	0,2933	5,1106	0,9662	0,2933	5,1106	0,9662

Time Series Splitting

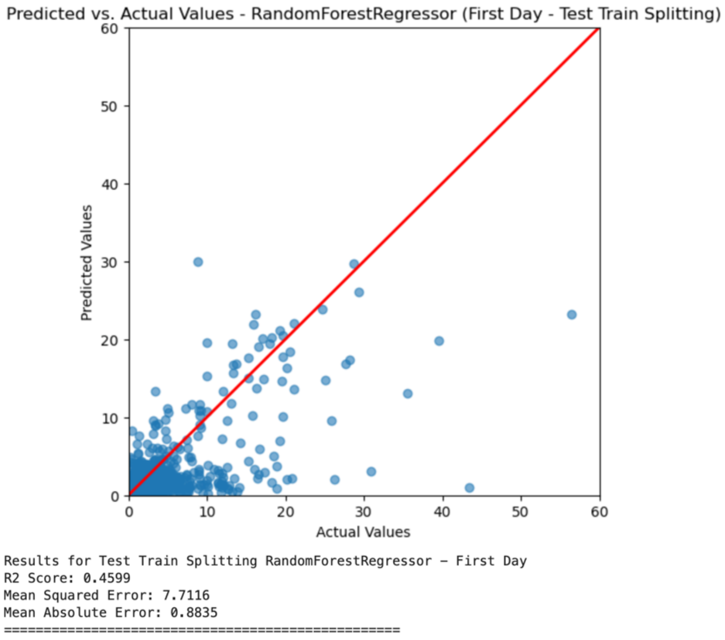
Regression Models	First Day			First Session		
	R2 Score	Mean Square	Mean Absolute	R2 Score	Mean Square	Mean Absolute
Random Forest Regressor	0,4762	4,7342	0,8785	0,2593	6,4876	1,0868
Decision Tree Regressor	0,3638	5,7503	0,9373	0,2282	6,7598	1,1198
LGBM Regressor	0,5066	4,4595	0,8712	0,2681	6,4101	1,1059

The regression models' plots are shown in the following pages:

Random Forest Regressor

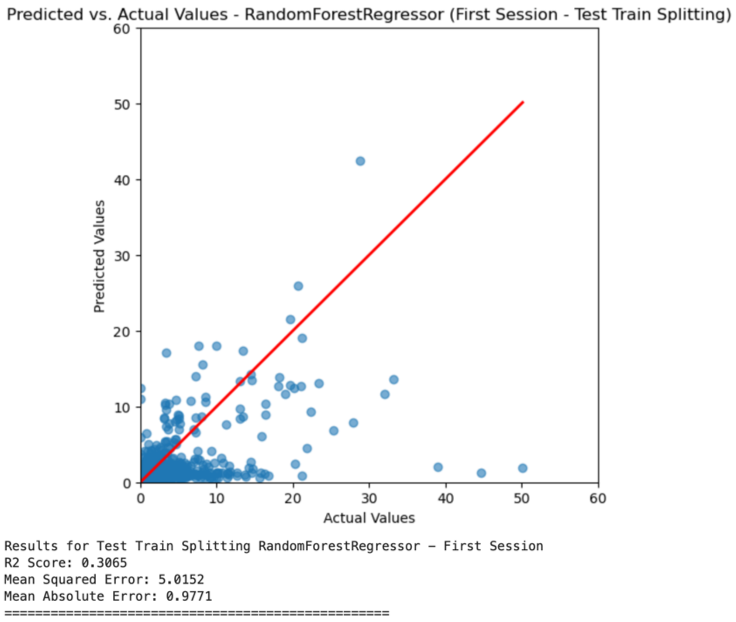
Test Train Splitting

First Day Data



```
Results for Test Train Splitting RandomForestRegressor – First Day
R2 Score: 0.4599
Mean Squared Error: 7.7116
Mean Absolute Error: 0.8835
==================================================
```

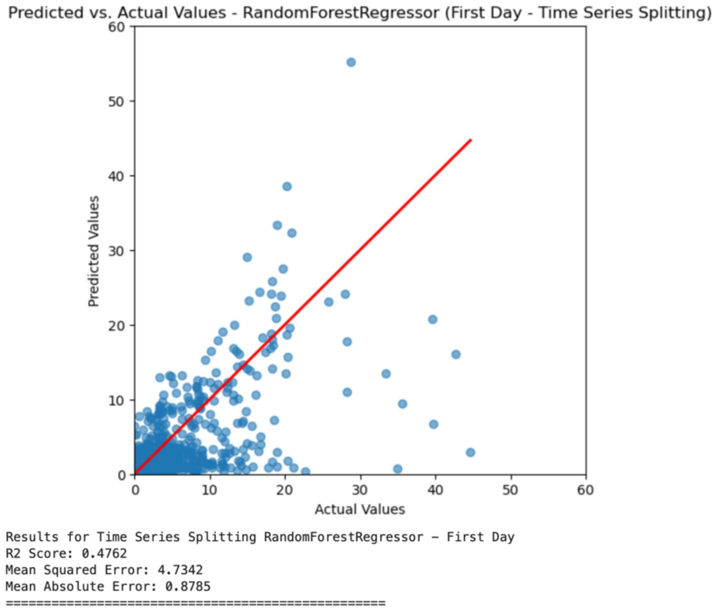
First Session Data



```
Results for Test Train Splitting RandomForestRegressor – First Session
R2 Score: 0.3065
Mean Squared Error: 5.0152
Mean Absolute Error: 0.9771
==================================================
```

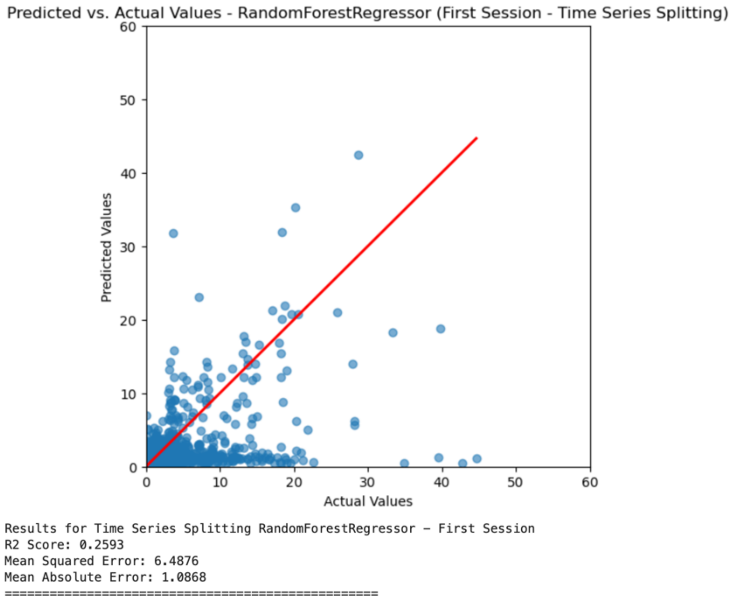
Time Series Splitting

First Day Data



```
Results for Time Series Splitting RandomForestRegressor – First Day
R2 Score: 0.4762
Mean Squared Error: 4.7342
Mean Absolute Error: 0.8785
==================================================
```

First Session Data

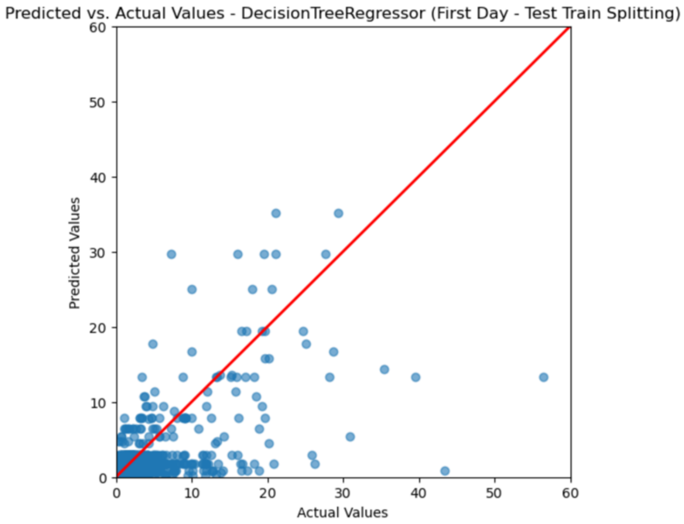


```
Results for Time Series Splitting RandomForestRegressor – First Session
R2 Score: 0.2593
Mean Squared Error: 6.4876
Mean Absolute Error: 1.0868
==================================================
```

Decision Tree Regressor

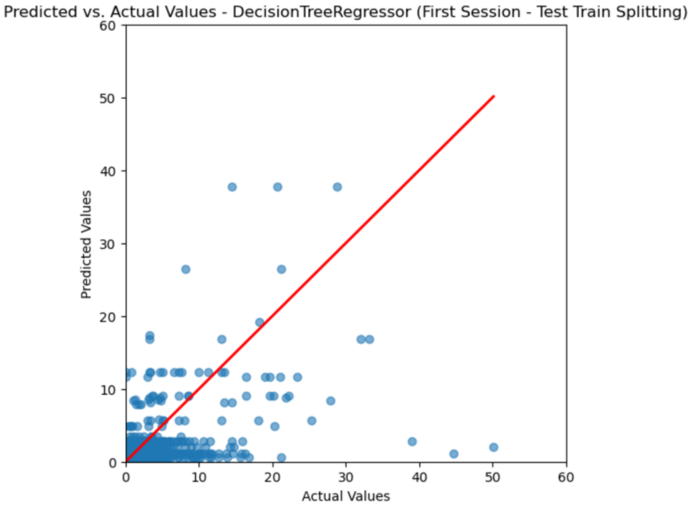
Test Train Splitting

First Day Data



```
Results for Test Train Splitting DecisionTreeRegressor – First Day
R2 Score: 0.4402
Mean Squared Error: 7.9923
Mean Absolute Error: 0.9422
```

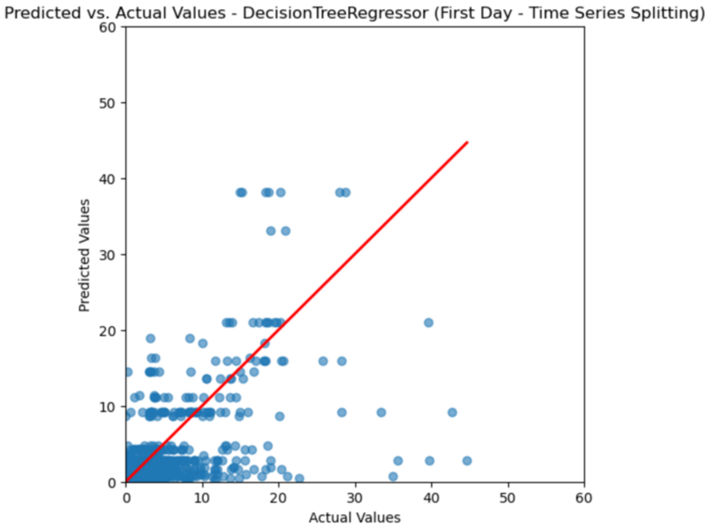
First Session Data



```
Results for Test Train Splitting DecisionTreeRegressor – First Session
R2 Score: 0.2462
Mean Squared Error: 5.4507
Mean Absolute Error: 1.0046
```

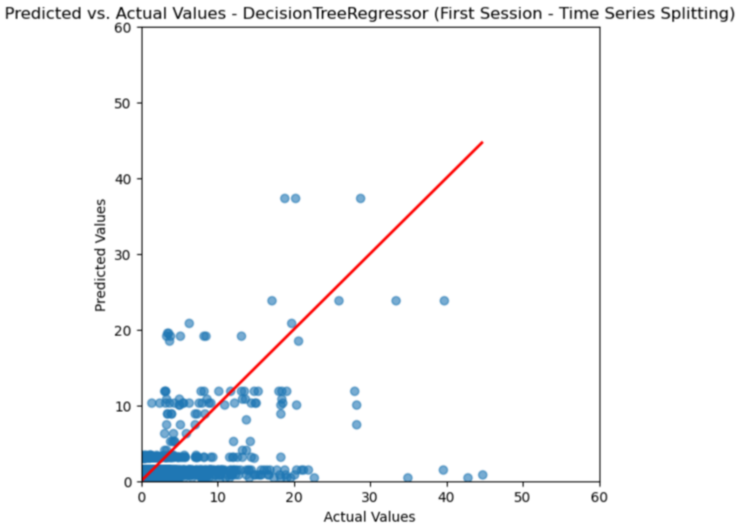
Time Series Splitting

First Day Data



```
Results for Time Series Splitting DecisionTreeRegressor – First Day
R2 Score: 0.3638
Mean Squared Error: 5.7503
Mean Absolute Error: 0.9373
```

First Session Data

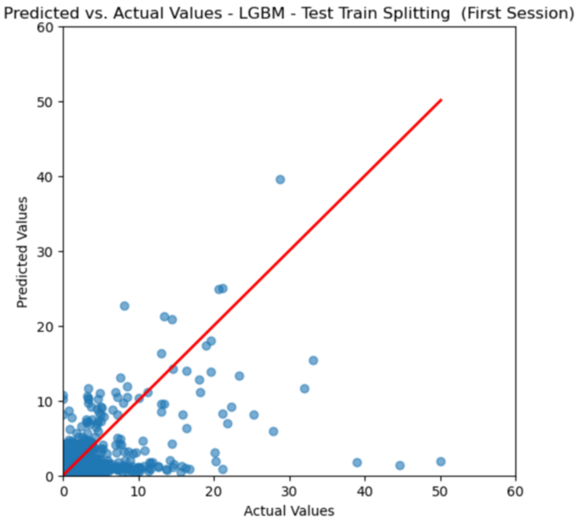


```
Results for Time Series Splitting DecisionTreeRegressor – First Session
R2 Score: 0.2282
Mean Squared Error: 6.7598
Mean Absolute Error: 1.1198
```

LGBM Regressor

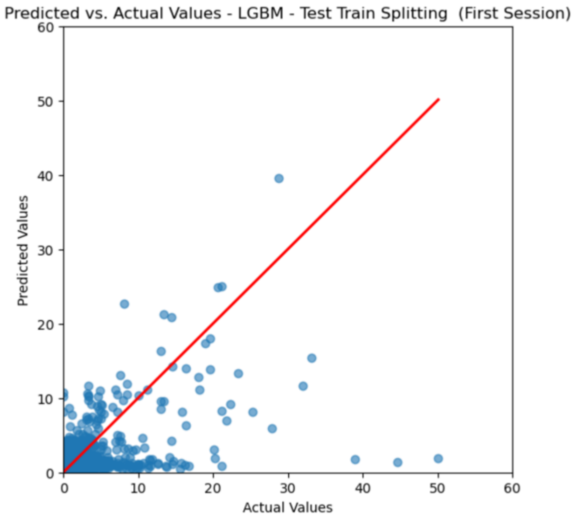
Test Train Splitting

First Day Data



```
Results for LGBM Regressor (Test Train Splitting) - First Session
R2 Score: 0.2933
Mean Squared Error: 5.1106
Mean Absolute Error: 0.9662
```

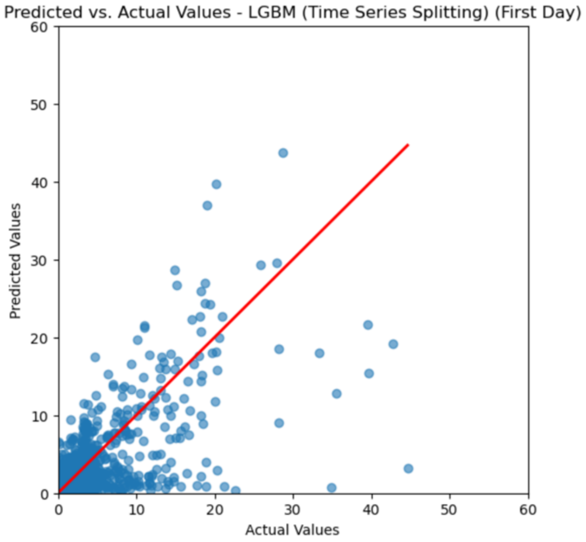
First Session Data



```
Results for LGBM Regressor (Test Train Splitting) - First Session
R2 Score: 0.2933
Mean Squared Error: 5.1106
Mean Absolute Error: 0.9662
```

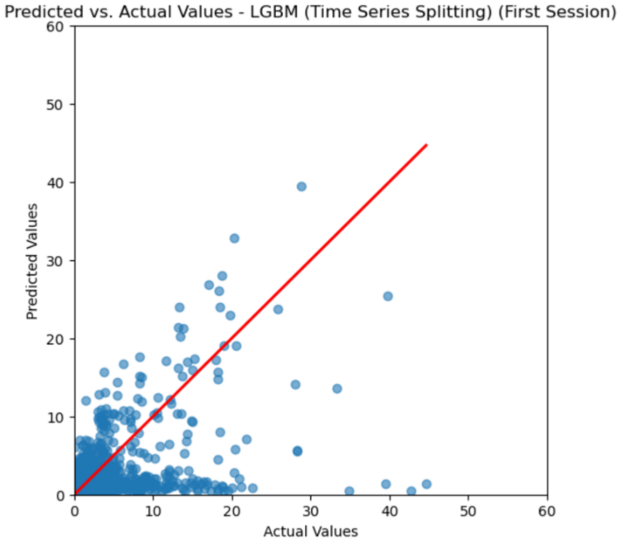
Time Series Splitting

First Day Data



```
Results for LGBM (Time Series Splitting) - First Day
R2 Score: 0.5066
Mean Squared Error: 4.4595
Mean Absolute Error: 0.8712
```

First Session Data



```
Results for LGBM (Time Series Splitting)- First Session
R2 Score: 0.2681
Mean Squared Error: 6.4101
Mean Absolute Error: 1.1059
```

Conclusion

The LGBM Regressor consistently performs best across both test-train and time-series splitting methods, with lower errors (MSE and MAE) and higher R2 scores, particularly excelling in time-series data. Random Forest Regressorperforms well in the test-train split but sees a drop in the time-series split, indicating it struggles with sequential data. Decision Tree Regressor performs the worst, with lower R2 scores and higher errors throughout, making it the least effective model overall.