shipping

December 6, 2023

1 Shipping Optimization Challenge

[48]:
```
# imports
import pandas as pd
import numpy as np
```

1.1 1. Load the data

[49]:
```
train_df = pd.read_csv('train_2_pr.csv')
```

1.2 2. Cleaning, EDA and Pre-processing

[50]:
```
# Automated EDA with Pandas Profiling

from ydata_profiling import ProfileReport

profile = ProfileReport(train_df, title='Pandas Profiling Report')

# show in notebook
profile.to_notebook_iframe()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[51]:
```
profile.to_file("EDA_report.html")
```

```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

[52]:
```
# get unique values
train_df.nunique()
```

[52]:
```
Unnamed: 0             5114
shipment_id            4805
```

```
send_timestamp          4804
pick_up_point              1
drop_off_point             2
source_country             1
destination_country        2
freight_cost            2000
gross_weight            1301
shipment_charges           5
shipment_mode              2
shipping_company           3
selected                   1
shipping_time           4315
dtype: int64
```

[53]:
```python
# drop the following columns: shipment_id, pickup_point, source_country,␣
 ↪selected because they only have 1 value
train_df.drop(['Unnamed: 0', 'shipment_id', 'pick_up_point', 'source_country',␣
 ↪'selected'], axis=1, inplace=True)

# print the df shape
print(train_df.shape)

# print the df head
train_df.head()
```

```
(5114, 9)
```

[53]:
```
        send_timestamp drop_off_point destination_country  freight_cost  \
0  2019-06-08 07:17:51              Y                  IN         88.61
1  2019-07-12 15:23:21              Y                  IN         85.65
2  2019-10-04 14:23:29              Y                  IN         86.22
3  2020-01-07 09:19:50              Y                  IN         94.43
4  2020-04-11 06:36:03              Y                  IN         94.24

   gross_weight  shipment_charges shipment_mode shipping_company  \
0         355.0              0.75           Air              SC3
1         105.0              0.90         Ocean              SC1
2         100.0              0.75           Air              SC3
3        1071.0              1.05           Air              SC2
4        2007.0              0.75           Air              SC3

   shipping_time
0        5.00741
1       21.41215
2        5.33692
3        5.14792
4        5.03067
```

Now, I will encode the categorical variables

[54]:
```
# Encode the following categorical variables into numeric ones: drop_off_point,
 ↪destination_country, shipment_mode, shipping_company

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

colums_to_encode = ['drop_off_point', 'destination_country', 'shipment_mode',
 ↪'shipping_company']

# Encode 'travel_from' and 'car_type' columns
for column in colums_to_encode:
    train_df[column + '_encoded'] = label_encoder.
 ↪fit_transform(train_df[column])
    # drop the column
    train_df.drop(column, axis=1, inplace=True)

# print the df shape
print(train_df.shape)

# print the df head
train_df.head()
```

```
(5114, 9)
```

[54]:
```
        send_timestamp  freight_cost  gross_weight  shipment_charges  \
0  2019-06-08 07:17:51         88.61         355.0              0.75
1  2019-07-12 15:23:21         85.65         105.0              0.90
2  2019-10-04 14:23:29         86.22         100.0              0.75
3  2020-01-07 09:19:50         94.43        1071.0              1.05
4  2020-04-11 06:36:03         94.24        2007.0              0.75

   shipping_time  drop_off_point_encoded  destination_country_encoded  \
0        5.00741                       1                            1
1       21.41215                       1                            1
2        5.33692                       1                            1
3        5.14792                       1                            1
4        5.03067                       1                            1

   shipment_mode_encoded  shipping_company_encoded
0                      0                         2
1                      1                         0
2                      0                         2
3                      0                         1
4                      0                         2
```

[55]:

```python
# check unique values
train_df.nunique()
```

[55]:

```
send_timestamp                4804
freight_cost                  2000
gross_weight                  1301
shipment_charges                 5
shipping_time                 4315
drop_off_point_encoded           2
destination_country_encoded      2
shipment_mode_encoded            2
shipping_company_encoded         3
dtype: int64
```

Check for empty cells

[56]:

```python
# check for null values
train_df.isnull().sum()
```

[56]:

```
send_timestamp                0
freight_cost                  0
gross_weight                  0
shipment_charges              0
shipping_time                 0
drop_off_point_encoded        0
destination_country_encoded   0
shipment_mode_encoded         0
shipping_company_encoded      0
dtype: int64
```

[57]:

```python
# Assuming 'send_timestamp' is in a train_dfFrame named 'train_df'
train_df['send_timestamp'] = pd.to_datetime(train_df['send_timestamp'])  #␣
↪Convert to datetime if not already in datetime format

# # get the day of year column from the send_timestamp column
train_df['day_of_year_sent'] = train_df['send_timestamp'].dt.dayofyear

# Extracting hour, minute, second
train_df['hour'] = train_df['send_timestamp'].dt.hour
train_df['minute'] = train_df['send_timestamp'].dt.minute
train_df['second'] = train_df['send_timestamp'].dt.second

# Applying cyclical encoding (sine and cosine transformations) for cyclical␣
↪patterns
train_df['hour_sin'] = np.sin(2 * np.pi * train_df['hour'] / 24.0)
train_df['hour_cos'] = np.cos(2 * np.pi * train_df['hour'] / 24.0)
```

```
train_df['minute_sin'] = np.sin(2 * np.pi * train_df['minute'] / 60.0)
train_df['minute_cos'] = np.cos(2 * np.pi * train_df['minute'] / 60.0)

train_df['second_sin'] = np.sin(2 * np.pi * train_df['second'] / 60.0)
train_df['second_cos'] = np.cos(2 * np.pi * train_df['second'] / 60.0)

# Drop the original 'send_timestamp' column, as well as the hour, minute,␣
↪second columns
train_df.drop('send_timestamp', axis=1, inplace=True)
train_df.drop('hour', axis=1, inplace=True)
train_df.drop('minute', axis=1, inplace=True)
train_df.drop('second', axis=1, inplace=True)

# print the df shape
print(train_df.shape)

# print the df head
train_df.head()
```

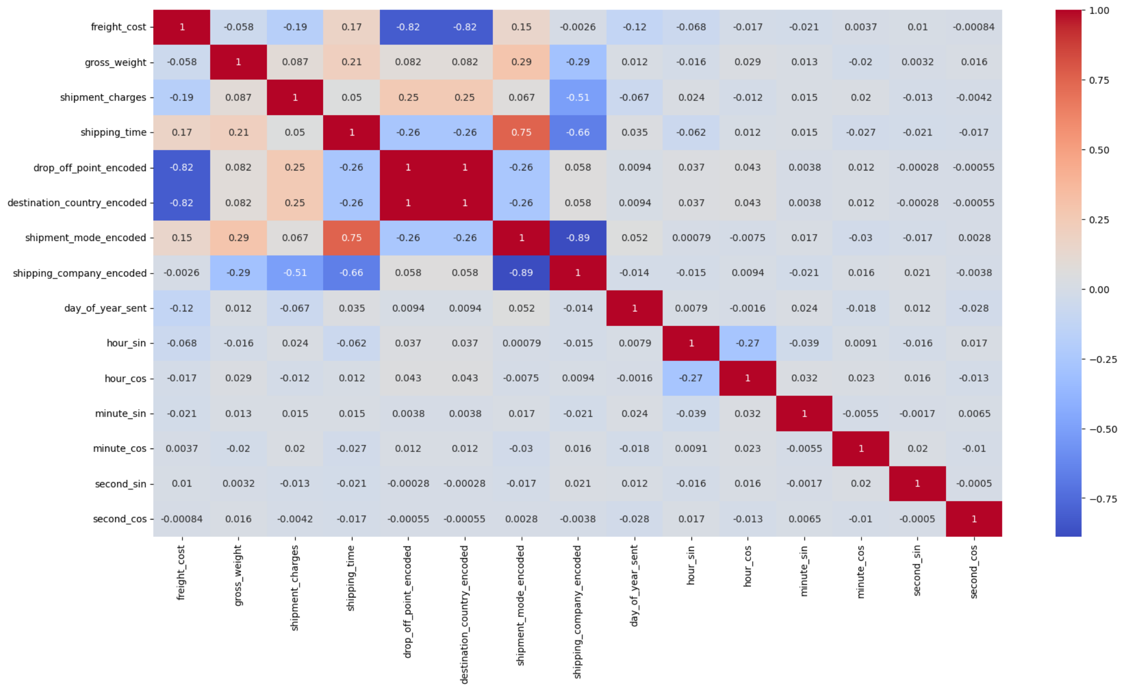
```
(5114, 15)
```

[57]:

```
   freight_cost  gross_weight  shipment_charges  shipping_time  \
0         88.61         355.0              0.75        5.00741
1         85.65         105.0              0.90       21.41215
2         86.22         100.0              0.75        5.33692
3         94.43        1071.0              1.05        5.14792
4         94.24        2007.0              0.75        5.03067

   drop_off_point_encoded  destination_country_encoded  shipment_mode_encoded  \
0                       1                            1                      0
1                       1                            1                      1
2                       1                            1                      0
3                       1                            1                      0
4                       1                            1                      0

   shipping_company_encoded  day_of_year_sent  hour_sin      hour_cos  \
0                         2               159  0.965926 -2.588190e-01
1                         0               193 -0.707107 -7.071068e-01
2                         2               277 -0.500000 -8.660254e-01
3                         1                 7  0.707107 -7.071068e-01
4                         2               102  1.000000  6.123234e-17

   minute_sin  minute_cos  second_sin  second_cos
0    0.978148   -0.207912   -0.809017    0.587785
1    0.669131   -0.743145    0.809017   -0.587785
2    0.669131   -0.743145    0.104528   -0.994522
3    0.913545   -0.406737   -0.866025    0.500000
```

```
4   -0.587785   -0.809017    0.309017    0.951057
```

[58]:
```python
# check correlation between variables using heatmap
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(20, 10))
sns.heatmap(train_df.corr(), annot=True, cmap='coolwarm')
plt.show()
```



[59]:
```python
# The following variables are highly correlated:
# 1. drop_off_point_encoded and destination_country_encoded
# 2. shipping_company_encoded and shipment_mode_encoded
# 3. freight_cost with destination_country_encoded and drop_off_point_encoded

# drop the following columns: drop_off_point_encoded, shipping_company_encoded, destination_country_encoded, hour_cos
train_df.drop(['drop_off_point_encoded',
               'hour_cos'
            #    'shipping_company_encoded',
            #    'destination_country_encoded'
               ], axis=1, inplace=True)

# print the df shape
print(train_df.shape)
```

```
# print the df head
train_df.head()
```

```
(5114, 13)
```

[59]:

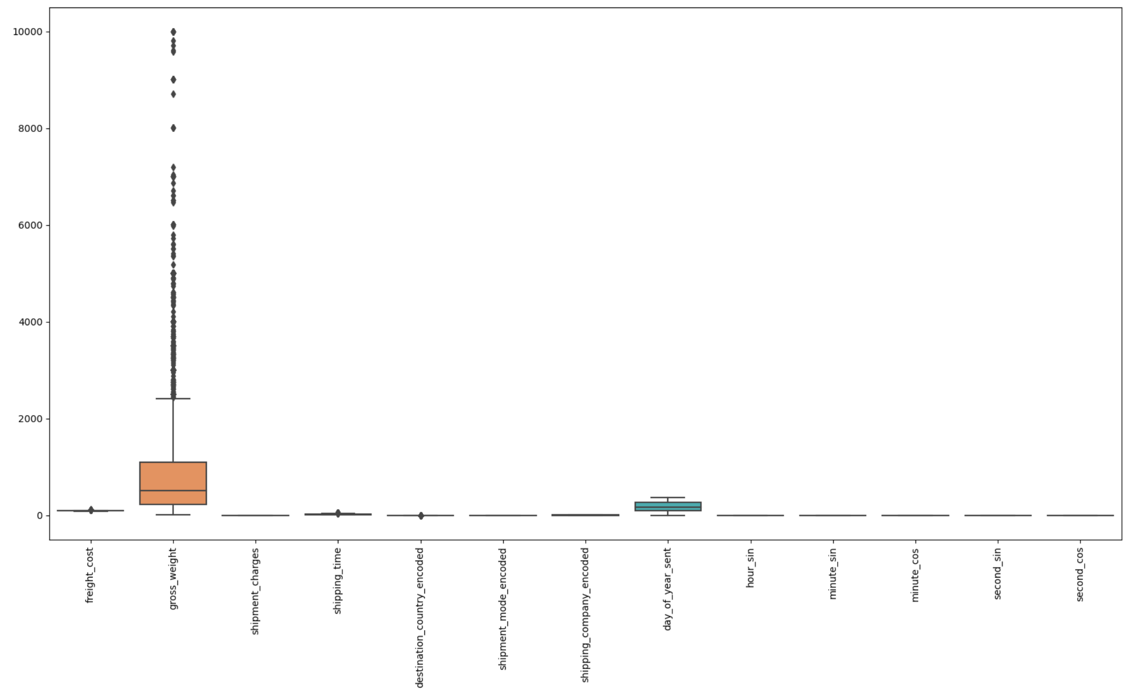
```
   freight_cost  gross_weight  shipment_charges  shipping_time  \
0         88.61         355.0              0.75        5.00741
1         85.65         105.0              0.90       21.41215
2         86.22         100.0              0.75        5.33692
3         94.43        1071.0              1.05        5.14792
4         94.24        2007.0              0.75        5.03067

   destination_country_encoded  shipment_mode_encoded  \
0                            1                      0
1                            1                      1
2                            1                      0
3                            1                      0
4                            1                      0

   shipping_company_encoded  day_of_year_sent  hour_sin  minute_sin  \
0                         2               159  0.965926    0.978148
1                         0               193 -0.707107    0.669131
2                         2               277 -0.500000    0.669131
3                         1                 7  0.707107    0.913545
4                         2               102  1.000000   -0.587785

   minute_cos  second_sin  second_cos
0   -0.207912   -0.809017    0.587785
1   -0.743145    0.809017   -0.587785
2   -0.743145    0.104528   -0.994522
3   -0.406737   -0.866025    0.500000
4   -0.809017    0.309017    0.951057
```

[60]:

```
# check for outliers
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(20, 10))
sns.boxplot(data=train_df)
plt.xticks(rotation=90)
plt.show()
```

```
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
```

```
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



[61]:
```
# drop gross_weight outliers
# train_df = train_df[train_df['gross_weight'] < 6000]

# drop gross_weight
# train_df.drop('gross_weight', axis=1, inplace=True)

# scale the gross_weight column and the freight_cost column and␣
↪send_day_of_year column
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
train_df['gross_weight'] = scaler.fit_transform(train_df[['gross_weight']])
train_df['freight_cost'] = scaler.fit_transform(train_df[['freight_cost']])
train_df['day_of_year_sent'] = scaler.
↪fit_transform(train_df[['day_of_year_sent']])

# print the df shape
print(train_df.shape)

# print the df head
train_df.head()
```

```
(5114, 13)
```

[61]:

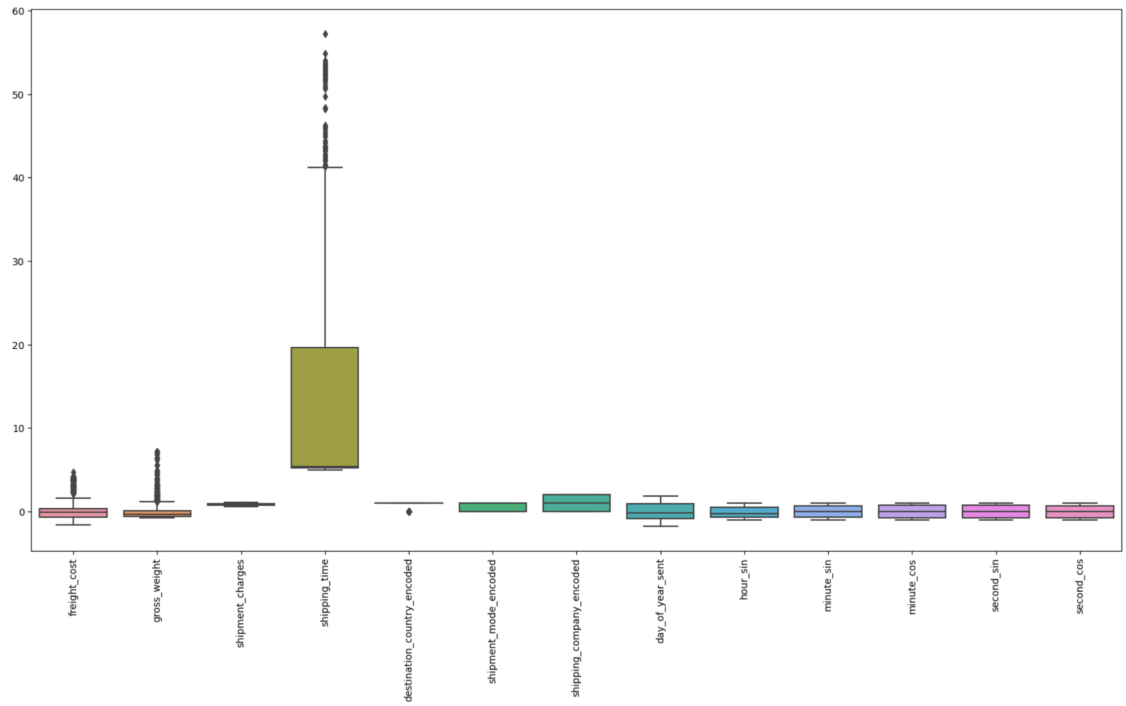
```
   freight_cost  gross_weight  shipment_charges  shipping_time  \
0     -0.502717     -0.473210              0.75        5.00741
1     -1.077047     -0.670686              0.90       21.41215
2     -0.966450     -0.674635              0.75        5.33692
3      0.626539      0.092360              1.05        5.14792
4      0.589673      0.831709              0.75        5.03067

   destination_country_encoded  shipment_mode_encoded  \
0                            1                      0
1                            1                      1
2                            1                      0
3                            1                      0
4                            1                      0

   shipping_company_encoded  day_of_year_sent  hour_sin  minute_sin  \
0                         2         -0.178622  0.965926    0.978148
1                         0          0.159260 -0.707107    0.669131
2                         2          0.994026 -0.500000    0.669131
3                         1         -1.689151  0.707107    0.913545
4                         2         -0.745070  1.000000   -0.587785

   minute_cos  second_sin  second_cos
0   -0.207912   -0.809017    0.587785
1   -0.743145    0.809017   -0.587785
2   -0.743145    0.104528   -0.994522
3   -0.406737   -0.866025    0.500000
4   -0.809017    0.309017    0.951057
```

[62]:

```
# check for outliers
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(20, 10))
sns.boxplot(data=train_df)
plt.xticks(rotation=90)
plt.show()
```

```
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
```

```
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/home/benson/.local/lib/python3.10/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



[63]:
```
# check for duplicate rows
train_df.duplicated().sum()
```

[63]: 0

[64]:
```
# make a copy of the df
final_df = train_df

# print the df head
final_df.head()
```

[64]:
```
   freight_cost  gross_weight  shipment_charges  shipping_time  \
0     -0.502717     -0.473210              0.75        5.00741
1     -1.077047     -0.670686              0.90       21.41215
2     -0.966450     -0.674635              0.75        5.33692
3      0.626539      0.092360              1.05        5.14792
4      0.589673      0.831709              0.75        5.03067

   destination_country_encoded  shipment_mode_encoded  \
0                            1                      0
1                            1                      1
2                            1                      0
3                            1                      0
4                            1                      0
```

```
   shipping_company_encoded  day_of_year_sent  hour_sin  minute_sin  \
0                         2         -0.178622  0.965926    0.978148
1                         0          0.159260 -0.707107    0.669131
2                         2          0.994026 -0.500000    0.669131
3                         1         -1.689151  0.707107    0.913545
4                         2         -0.745070  1.000000   -0.587785

   minute_cos  second_sin  second_cos
0   -0.207912   -0.809017    0.587785
1   -0.743145    0.809017   -0.587785
2   -0.743145    0.104528   -0.994522
3   -0.406737   -0.866025    0.500000
4   -0.809017    0.309017    0.951057
```

1.3 Model Training

[65]:
```
from sklearn.model_selection import train_test_split

# Splitting the data into train and test sets (80% train, 20% test)
X = final_df.drop('shipping_time', axis=1)  # Features
y = final_df['shipping_time']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[126]:
```
# SVM model

from sklearn.svm import SVR

# Initializing the SVM model
svm_model = SVR(kernel='rbf')  # You can experiment with different kernels␣
↪(linear, rbf, poly)

# Training the SVM model
svm_model.fit(X_train, y_train)
```

[126]: SVR()

[127]:
```
from sklearn.metrics import mean_squared_error

# Predicting with SVM model
svm_predictions = svm_model.predict(X_test)

# Evaluating model performance
svm_mse = mean_squared_error(y_test, svm_predictions)
svm_rmse = np.sqrt(svm_mse)
```

```
print(f"SVM Mean Squared Error (MSE): {svm_mse}")
print(f"SVM Root Mean Squared Error (RMSE): {svm_rmse}")
```

```
SVM Mean Squared Error (MSE): 50.02683225365311
SVM Root Mean Squared Error (RMSE): 7.07296488423724
```

[68]:
```
# Hyperparameter Tuning with GridSearchCV
from sklearn.model_selection import GridSearchCV

# Define the parameter grid to search
param_grid = {
    'C': [0.01, 5, 10],  # Regularization parameter
    'gamma': [0.001, 0.01, 1],  # Kernel coefficient for RBF
    'kernel': ['rbf', 'sigmoid']  # Kernel type
}

# Initialize the SVM model
svm_model = SVR()

# Create GridSearchCV
grid_search = GridSearchCV(estimator=svm_model, param_grid=param_grid,␣
 ↪scoring='neg_mean_squared_error', cv=5)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Predict using the best model
best_svm_model = grid_search.best_estimator_
svm_predictions = best_svm_model.predict(X_test)

# Evaluate model performance
svm_mse = mean_squared_error(y_test, svm_predictions)
svm_rmse = np.sqrt(svm_mse)
print(f"Improved SVM Mean Squared Error (MSE): {svm_mse}")
print(f"Improved SVM Root Mean Squared Error (RMSE): {svm_rmse}")
```

```
Best Parameters: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
Improved SVM Mean Squared Error (MSE): 48.863685435983925
Improved SVM Root Mean Squared Error (RMSE): 6.990256464249644
```

SVM Tuning

[154]:
```
# SVM model
```

```
from sklearn.svm import SVR

# Initializing the SVM model
final_svm_model = SVR(kernel='rbf', C= 10, gamma=0.01)  # You can experiment␣
↪with different kernels (linear, rbf, poly)

# Training the SVM model
final_svm_model.fit(X_train, y_train)
```

[154]: SVR(C=10, gamma=0.01)

[157]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Predicting with SVM model
svm_predictions = final_svm_model.predict(X_test)

# Evaluating model performance
svm_mse = mean_squared_error(y_test, svm_predictions)
svm_mae = mean_absolute_error(y_test, svm_predictions)
svm_rmse = np.sqrt(svm_mse)
svm_r2 = r2_score(y_test, svm_predictions)
print(f"SVM Mean Squared Error (MSE): {svm_mse}")
print(f"SVM Root Mean Squared Error (RMSE): {svm_rmse}")
print(f"SVM Mean Absolute Error (MAE): {svm_mae}")
print(f"SVM R2 Score: {svm_r2}")
```

```
SVM Mean Squared Error (MSE): 48.863685435983925
SVM Root Mean Squared Error (RMSE): 6.990256464249644
SVM Mean Absolute Error (MAE): 3.9636555278829353
SVM R2 Score: 0.5548585161654551
```

[71]:
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import reciprocal, uniform

# Define the parameter distribution to search
param_dist = {
    'C': reciprocal(0.1, 15),  # Random distribution for 'C'
    'gamma': reciprocal(0.01, 10),  # Random distribution for 'gamma'
    'kernel': ['rbf', 'linear']  # Kernel type
}

# Initialize the SVM model
svm_model = SVR()

# Create RandomizedSearchCV
```

```
random_search = RandomizedSearchCV(estimator=svm_model,␣
↪param_distributions=param_dist, scoring='neg_mean_squared_error', cv=5,␣
↪n_iter=10, random_state=42)

# Fit the randomized search to the data
random_search.fit(X_train, y_train)

# Get the best parameters
best_params = random_search.best_params_
print("Best Parameters:", best_params)

# Predict using the best model
best_svm_model = random_search.best_estimator_
svm_predictions = best_svm_model.predict(X_test)

# Evaluate model performance
svm_mse = mean_squared_error(y_test, svm_predictions)
svm_rmse = np.sqrt(svm_mse)
print(f"Improved SVM Mean Squared Error (MSE): {svm_mse}")
print(f"Improved SVM Root Mean Squared Error (RMSE): {svm_rmse}")
```

```
Best Parameters: {'C': 13.151669089586505, 'gamma': 0.0499245341692398,
'kernel': 'linear'}
Improved SVM Mean Squared Error (MSE): 49.18693105802111
Improved SVM Root Mean Squared Error (RMSE): 7.013339508252906
```

Now we will make the Neural Network

[158]:
```
# make a NN
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# define the model
model = keras.Sequential([
    layers.Dense(8, activation='relu', input_shape=[12]),  # Update input shape␣
↪to (None, 5)
    layers.Dense(12, activation='relu'),
    layers.Dense(1)
])

# compile the model
model.compile(
    optimizer='adam',
    loss='mae',
    metrics=['mae', 'mse']
)
```

```python
# fit the model
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    batch_size=32,
    epochs=100
)

# evaluate the model
model.evaluate(X_test, y_test)
```

```
Epoch 1/100
128/128 [==============================] - 6s 16ms/step - loss: 9.3707 - mae:
9.3707 - mse: 186.7384 - val_loss: 7.2157 - val_mae: 7.2157 - val_mse: 143.1306
Epoch 2/100
128/128 [==============================] - 1s 11ms/step - loss: 6.1698 - mae:
6.1698 - mse: 110.5186 - val_loss: 5.6409 - val_mae: 5.6409 - val_mse: 100.3298
Epoch 3/100
128/128 [==============================] - 1s 5ms/step - loss: 5.1286 - mae:
5.1286 - mse: 79.4166 - val_loss: 5.1130 - val_mae: 5.1130 - val_mse: 78.2537
Epoch 4/100
128/128 [==============================] - 0s 4ms/step - loss: 4.7211 - mae:
4.7211 - mse: 64.0088 - val_loss: 4.8137 - val_mae: 4.8137 - val_mse: 68.1921
Epoch 5/100
128/128 [==============================] - 1s 4ms/step - loss: 4.4249 - mae:
4.4249 - mse: 56.5032 - val_loss: 4.5710 - val_mae: 4.5710 - val_mse: 62.0131
Epoch 6/100
128/128 [==============================] - 0s 3ms/step - loss: 4.1644 - mae:
4.1644 - mse: 51.9466 - val_loss: 4.3317 - val_mae: 4.3317 - val_mse: 57.6848
Epoch 7/100
128/128 [==============================] - 1s 6ms/step - loss: 3.9551 - mae:
3.9551 - mse: 48.7768 - val_loss: 4.1494 - val_mae: 4.1494 - val_mse: 54.6501
Epoch 8/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8393 - mae:
3.8393 - mse: 47.1771 - val_loss: 4.0893 - val_mae: 4.0893 - val_mse: 53.7440
Epoch 9/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8057 - mae:
3.8057 - mse: 46.7735 - val_loss: 4.0674 - val_mae: 4.0674 - val_mse: 53.5184
Epoch 10/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7901 - mae:
3.7901 - mse: 46.5100 - val_loss: 4.0513 - val_mae: 4.0513 - val_mse: 52.9892
Epoch 11/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7822 - mae:
3.7822 - mse: 46.2688 - val_loss: 4.0511 - val_mae: 4.0511 - val_mse: 52.7700
Epoch 12/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7747 - mae:
3.7747 - mse: 46.1572 - val_loss: 4.0445 - val_mae: 4.0445 - val_mse: 52.4318
```

```
Epoch 13/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7680 - mae:
3.7680 - mse: 45.8401 - val_loss: 4.0307 - val_mae: 4.0307 - val_mse: 52.2148
Epoch 14/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7650 - mae:
3.7650 - mse: 45.7196 - val_loss: 4.0459 - val_mae: 4.0459 - val_mse: 51.9171
Epoch 15/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7643 - mae:
3.7643 - mse: 45.4820 - val_loss: 4.0223 - val_mae: 4.0223 - val_mse: 51.8842
Epoch 16/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7573 - mae:
3.7573 - mse: 45.5800 - val_loss: 4.0248 - val_mae: 4.0248 - val_mse: 51.5452
Epoch 17/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7549 - mae:
3.7549 - mse: 45.2501 - val_loss: 4.0168 - val_mae: 4.0168 - val_mse: 51.5832
Epoch 18/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7572 - mae:
3.7572 - mse: 45.3318 - val_loss: 4.0140 - val_mae: 4.0140 - val_mse: 51.5577
Epoch 19/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7524 - mae:
3.7524 - mse: 45.2714 - val_loss: 4.0164 - val_mae: 4.0164 - val_mse: 50.8903
Epoch 20/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7518 - mae:
3.7518 - mse: 44.9871 - val_loss: 4.0126 - val_mae: 4.0126 - val_mse: 51.1239
Epoch 21/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7500 - mae:
3.7500 - mse: 44.9617 - val_loss: 4.0053 - val_mae: 4.0053 - val_mse: 50.9254
Epoch 22/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7494 - mae:
3.7494 - mse: 44.8584 - val_loss: 4.0002 - val_mae: 4.0002 - val_mse: 50.7710
Epoch 23/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7495 - mae:
3.7495 - mse: 44.9118 - val_loss: 4.0021 - val_mae: 4.0021 - val_mse: 50.6891
Epoch 24/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7435 - mae:
3.7435 - mse: 44.7818 - val_loss: 4.0003 - val_mae: 4.0003 - val_mse: 50.6017
Epoch 25/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7499 - mae:
3.7499 - mse: 44.7605 - val_loss: 4.0013 - val_mae: 4.0013 - val_mse: 50.6751
Epoch 26/100
128/128 [==============================] - 1s 4ms/step - loss: 3.7464 - mae:
3.7464 - mse: 44.6941 - val_loss: 4.0007 - val_mae: 4.0007 - val_mse: 50.4490
Epoch 27/100
128/128 [==============================] - 1s 6ms/step - loss: 3.7440 - mae:
3.7440 - mse: 44.6099 - val_loss: 3.9916 - val_mae: 3.9916 - val_mse: 50.3296
Epoch 28/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7421 - mae:
3.7421 - mse: 44.6075 - val_loss: 3.9955 - val_mae: 3.9955 - val_mse: 50.2963
```

```
Epoch 29/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7438 - mae:
3.7438 - mse: 44.6021 - val_loss: 3.9974 - val_mae: 3.9974 - val_mse: 50.2013
Epoch 30/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7425 - mae:
3.7425 - mse: 44.4846 - val_loss: 3.9931 - val_mae: 3.9931 - val_mse: 50.2309
Epoch 31/100
128/128 [==============================] - 1s 7ms/step - loss: 3.7414 - mae:
3.7414 - mse: 44.5142 - val_loss: 3.9862 - val_mae: 3.9862 - val_mse: 50.0572
Epoch 32/100
128/128 [==============================] - 1s 6ms/step - loss: 3.7396 - mae:
3.7396 - mse: 44.3648 - val_loss: 3.9926 - val_mae: 3.9926 - val_mse: 49.9719
Epoch 33/100
128/128 [==============================] - 1s 6ms/step - loss: 3.7410 - mae:
3.7410 - mse: 44.4572 - val_loss: 3.9846 - val_mae: 3.9846 - val_mse: 49.7967
Epoch 34/100
128/128 [==============================] - 1s 8ms/step - loss: 3.7408 - mae:
3.7408 - mse: 44.3720 - val_loss: 3.9815 - val_mae: 3.9815 - val_mse: 49.7933
Epoch 35/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7427 - mae:
3.7427 - mse: 44.3384 - val_loss: 3.9904 - val_mae: 3.9904 - val_mse: 49.9281
Epoch 36/100
128/128 [==============================] - 1s 6ms/step - loss: 3.7402 - mae:
3.7402 - mse: 44.4118 - val_loss: 3.9861 - val_mae: 3.9861 - val_mse: 49.9655
Epoch 37/100
128/128 [==============================] - 0s 4ms/step - loss: 3.7407 - mae:
3.7407 - mse: 44.3930 - val_loss: 3.9842 - val_mae: 3.9842 - val_mse: 49.7866
Epoch 38/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7384 - mae:
3.7384 - mse: 44.3483 - val_loss: 3.9891 - val_mae: 3.9891 - val_mse: 49.7558
Epoch 39/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7395 - mae:
3.7395 - mse: 44.3985 - val_loss: 3.9922 - val_mae: 3.9922 - val_mse: 50.0177
Epoch 40/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7406 - mae:
3.7406 - mse: 44.3314 - val_loss: 3.9863 - val_mae: 3.9863 - val_mse: 49.6692
Epoch 41/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7375 - mae:
3.7375 - mse: 44.2174 - val_loss: 3.9875 - val_mae: 3.9875 - val_mse: 49.9667
Epoch 42/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7402 - mae:
3.7402 - mse: 44.2853 - val_loss: 3.9823 - val_mae: 3.9823 - val_mse: 49.7902
Epoch 43/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7366 - mae:
3.7366 - mse: 44.2810 - val_loss: 3.9815 - val_mae: 3.9815 - val_mse: 49.9094
Epoch 44/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7406 - mae:
3.7406 - mse: 44.3565 - val_loss: 3.9780 - val_mae: 3.9780 - val_mse: 49.7052
```

```
Epoch 45/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7369 - mae:
3.7369 - mse: 44.2554 - val_loss: 3.9820 - val_mae: 3.9820 - val_mse: 49.8001
Epoch 46/100
128/128 [==============================] - 0s 4ms/step - loss: 3.7397 - mae:
3.7397 - mse: 44.2699 - val_loss: 3.9775 - val_mae: 3.9775 - val_mse: 49.5881
Epoch 47/100
128/128 [==============================] - 1s 6ms/step - loss: 3.7379 - mae:
3.7379 - mse: 44.2315 - val_loss: 3.9836 - val_mae: 3.9836 - val_mse: 49.8226
Epoch 48/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7371 - mae:
3.7371 - mse: 44.2927 - val_loss: 3.9770 - val_mae: 3.9770 - val_mse: 49.7140
Epoch 49/100
128/128 [==============================] - 0s 4ms/step - loss: 3.7362 - mae:
3.7362 - mse: 44.2779 - val_loss: 3.9803 - val_mae: 3.9803 - val_mse: 49.5951
Epoch 50/100
128/128 [==============================] - 1s 6ms/step - loss: 3.7358 - mae:
3.7358 - mse: 44.1429 - val_loss: 3.9822 - val_mae: 3.9822 - val_mse: 49.5776
Epoch 51/100
128/128 [==============================] - 1s 4ms/step - loss: 3.7373 - mae:
3.7373 - mse: 44.2594 - val_loss: 3.9883 - val_mae: 3.9883 - val_mse: 49.5912
Epoch 52/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7363 - mae:
3.7363 - mse: 44.2007 - val_loss: 3.9769 - val_mae: 3.9769 - val_mse: 49.6229
Epoch 53/100
128/128 [==============================] - 1s 4ms/step - loss: 3.7377 - mae:
3.7377 - mse: 44.1201 - val_loss: 3.9792 - val_mae: 3.9792 - val_mse: 49.4857
Epoch 54/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7362 - mae:
3.7362 - mse: 44.2189 - val_loss: 3.9804 - val_mae: 3.9804 - val_mse: 49.3654
Epoch 55/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7360 - mae:
3.7360 - mse: 44.0979 - val_loss: 3.9763 - val_mae: 3.9763 - val_mse: 49.4132
Epoch 56/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7354 - mae:
3.7354 - mse: 44.0684 - val_loss: 3.9756 - val_mae: 3.9756 - val_mse: 49.6183
Epoch 57/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7345 - mae:
3.7345 - mse: 44.0841 - val_loss: 3.9833 - val_mae: 3.9833 - val_mse: 49.6553
Epoch 58/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7334 - mae:
3.7334 - mse: 44.0948 - val_loss: 3.9848 - val_mae: 3.9848 - val_mse: 49.4361
Epoch 59/100
128/128 [==============================] - 0s 4ms/step - loss: 3.7346 - mae:
3.7346 - mse: 44.0443 - val_loss: 3.9771 - val_mae: 3.9771 - val_mse: 49.4678
Epoch 60/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7366 - mae:
3.7366 - mse: 44.0991 - val_loss: 3.9935 - val_mae: 3.9935 - val_mse: 49.4206
```

```
Epoch 61/100
128/128 [==============================] - 0s 4ms/step - loss: 3.7358 - mae:
3.7358 - mse: 44.1940 - val_loss: 3.9754 - val_mae: 3.9754 - val_mse: 49.4961
Epoch 62/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7336 - mae:
3.7336 - mse: 44.0187 - val_loss: 3.9777 - val_mae: 3.9777 - val_mse: 49.4522
Epoch 63/100
128/128 [==============================] - 1s 7ms/step - loss: 3.7334 - mae:
3.7334 - mse: 44.1045 - val_loss: 3.9772 - val_mae: 3.9772 - val_mse: 49.5349
Epoch 64/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7322 - mae:
3.7322 - mse: 44.1368 - val_loss: 3.9776 - val_mae: 3.9776 - val_mse: 49.3875
Epoch 65/100
128/128 [==============================] - 1s 5ms/step - loss: 3.7362 - mae:
3.7362 - mse: 43.9921 - val_loss: 3.9768 - val_mae: 3.9768 - val_mse: 49.3884
Epoch 66/100
128/128 [==============================] - 1s 4ms/step - loss: 3.7352 - mae:
3.7352 - mse: 44.0888 - val_loss: 3.9746 - val_mae: 3.9746 - val_mse: 49.3633
Epoch 67/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7373 - mae:
3.7373 - mse: 44.0291 - val_loss: 3.9869 - val_mae: 3.9869 - val_mse: 49.6842
Epoch 68/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7365 - mae:
3.7365 - mse: 44.1669 - val_loss: 3.9806 - val_mae: 3.9806 - val_mse: 49.5482
Epoch 69/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7358 - mae:
3.7358 - mse: 44.1010 - val_loss: 3.9810 - val_mae: 3.9810 - val_mse: 49.4462
Epoch 70/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7337 - mae:
3.7337 - mse: 44.1879 - val_loss: 3.9713 - val_mae: 3.9713 - val_mse: 49.2701
Epoch 71/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7326 - mae:
3.7326 - mse: 44.0625 - val_loss: 3.9780 - val_mae: 3.9780 - val_mse: 49.2433
Epoch 72/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7348 - mae:
3.7348 - mse: 44.0434 - val_loss: 3.9769 - val_mae: 3.9769 - val_mse: 49.2634
Epoch 73/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7328 - mae:
3.7328 - mse: 44.1616 - val_loss: 3.9781 - val_mae: 3.9781 - val_mse: 49.3776
Epoch 74/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7316 - mae:
3.7316 - mse: 43.9921 - val_loss: 3.9787 - val_mae: 3.9787 - val_mse: 49.3635
Epoch 75/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7332 - mae:
3.7332 - mse: 44.0897 - val_loss: 3.9727 - val_mae: 3.9727 - val_mse: 49.2519
Epoch 76/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7353 - mae:
3.7353 - mse: 44.0538 - val_loss: 3.9733 - val_mae: 3.9733 - val_mse: 49.2191
```

```
Epoch 77/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7347 - mae:
3.7347 - mse: 43.9907 - val_loss: 3.9723 - val_mae: 3.9723 - val_mse: 49.3185
Epoch 78/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7308 - mae:
3.7308 - mse: 44.0565 - val_loss: 3.9703 - val_mae: 3.9703 - val_mse: 49.1779
Epoch 79/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7324 - mae:
3.7324 - mse: 43.9284 - val_loss: 3.9720 - val_mae: 3.9720 - val_mse: 49.2529
Epoch 80/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7329 - mae:
3.7329 - mse: 44.0352 - val_loss: 3.9741 - val_mae: 3.9741 - val_mse: 49.2025
Epoch 81/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7338 - mae:
3.7338 - mse: 43.9971 - val_loss: 3.9758 - val_mae: 3.9758 - val_mse: 49.4227
Epoch 82/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7341 - mae:
3.7341 - mse: 43.9679 - val_loss: 3.9769 - val_mae: 3.9769 - val_mse: 49.2188
Epoch 83/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7308 - mae:
3.7308 - mse: 43.9515 - val_loss: 3.9734 - val_mae: 3.9734 - val_mse: 49.0897
Epoch 84/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7309 - mae:
3.7309 - mse: 44.0050 - val_loss: 3.9734 - val_mae: 3.9734 - val_mse: 49.2872
Epoch 85/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7349 - mae:
3.7349 - mse: 44.0547 - val_loss: 3.9692 - val_mae: 3.9692 - val_mse: 49.1798
Epoch 86/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7332 - mae:
3.7332 - mse: 44.0129 - val_loss: 3.9802 - val_mae: 3.9802 - val_mse: 49.2422
Epoch 87/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7319 - mae:
3.7319 - mse: 43.9071 - val_loss: 3.9706 - val_mae: 3.9706 - val_mse: 49.3043
Epoch 88/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7305 - mae:
3.7305 - mse: 43.9898 - val_loss: 3.9768 - val_mae: 3.9768 - val_mse: 49.3622
Epoch 89/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7355 - mae:
3.7355 - mse: 44.0862 - val_loss: 3.9702 - val_mae: 3.9702 - val_mse: 49.2947
Epoch 90/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7286 - mae:
3.7286 - mse: 44.0128 - val_loss: 3.9728 - val_mae: 3.9728 - val_mse: 49.3856
Epoch 91/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7302 - mae:
3.7302 - mse: 44.0002 - val_loss: 3.9754 - val_mae: 3.9754 - val_mse: 49.1842
Epoch 92/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7310 - mae:
3.7310 - mse: 43.9309 - val_loss: 3.9750 - val_mae: 3.9750 - val_mse: 49.4826
```

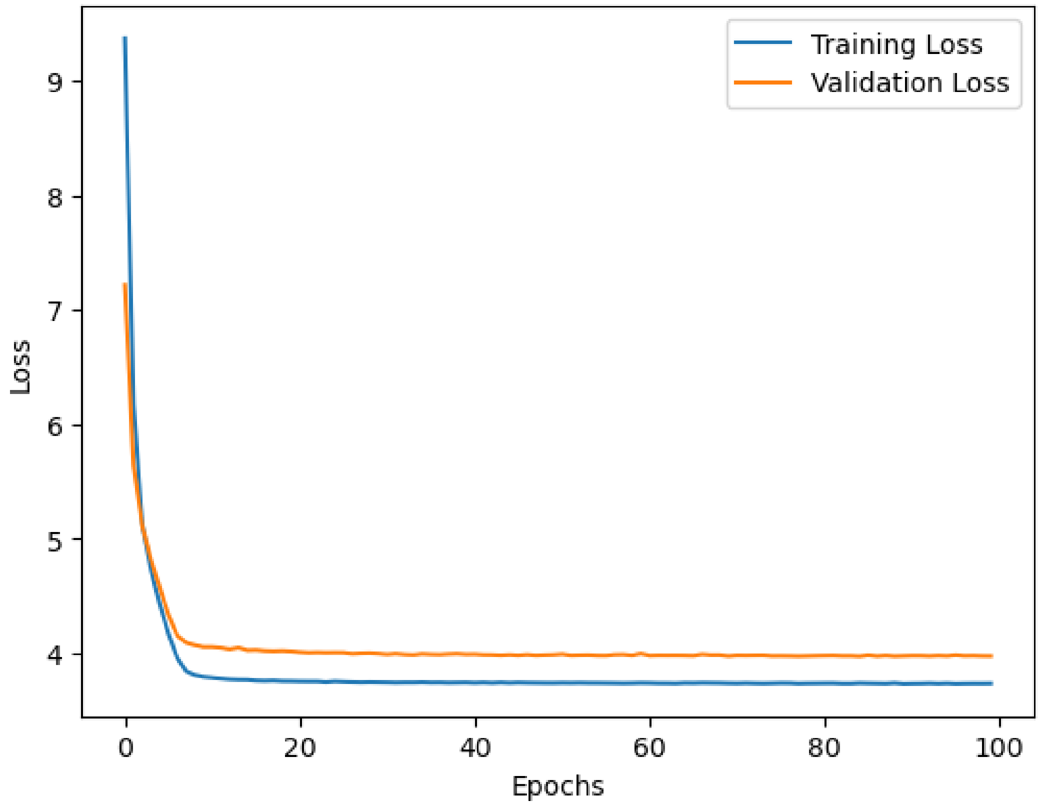
```
Epoch 93/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7331 - mae:
3.7331 - mse: 43.9922 - val_loss: 3.9718 - val_mae: 3.9718 - val_mse: 49.3655
Epoch 94/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7302 - mae:
3.7302 - mse: 43.9950 - val_loss: 3.9762 - val_mae: 3.9762 - val_mse: 49.4441
Epoch 95/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7328 - mae:
3.7328 - mse: 44.0504 - val_loss: 3.9727 - val_mae: 3.9727 - val_mse: 49.2943
Epoch 96/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7291 - mae:
3.7291 - mse: 43.8290 - val_loss: 3.9808 - val_mae: 3.9808 - val_mse: 49.2903
Epoch 97/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7305 - mae:
3.7305 - mse: 43.8736 - val_loss: 3.9739 - val_mae: 3.9739 - val_mse: 49.4393
Epoch 98/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7313 - mae:
3.7313 - mse: 44.0338 - val_loss: 3.9752 - val_mae: 3.9752 - val_mse: 49.4057
Epoch 99/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7310 - mae:
3.7310 - mse: 43.9587 - val_loss: 3.9723 - val_mae: 3.9723 - val_mse: 49.3190
Epoch 100/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7318 - mae:
3.7318 - mse: 44.0191 - val_loss: 3.9715 - val_mae: 3.9715 - val_mse: 49.4178
32/32 [==============================] - 0s 1ms/step - loss: 3.9715 - mae:
3.9715 - mse: 49.4178
```

[158]:
```
[3.9715282917022705, 3.9715282917022705, 49.41778564453125]
```

[159]:
```
import matplotlib.pyplot as plt

# Evaluating model performance
mse = model.evaluate(X_test, y_test)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, model.predict(X_test))
r2_score = r2_score(y_test, model.predict(X_test))
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"R2 Score: {r2_score}")

# Plotting training/validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
plt.show()
```

```
32/32 [==============================] - 0s 2ms/step - loss: 3.9715 - mae:
3.9715 - mse: 49.4178
32/32 [==============================] - 0s 1ms/step
32/32 [==============================] - 0s 1ms/step
Mean Squared Error (MSE): [3.9715282917022705, 3.9715282917022705,
49.41778564453125]
Root Mean Squared Error (RMSE): [1.99286936 1.99286936 7.02977849]
Mean Absolute Error (MAE): 3.971528664249185
R2 Score: 0.5498107371649763
```



Neural Network tuning

1. Adjust Model Architecture (Adding Dropout)

[74]:
```
# make a NN
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# ! adjust the layers as need be
```

```
model = keras.Sequential([
    layers.Dense(8, activation='relu', input_shape=[12]),  # Update input shape␣
 ↪to match your feature count
    layers.Dropout(0.2),  # Adding dropout to the first hidden layer
    layers.Dense(12, activation='relu'),
    layers.Dense(1)
])


# compile the model
model.compile(
    optimizer='adam',
    loss='mae',
    metrics=['mae', 'mse']
)

# fit the model
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    batch_size=32,
    epochs=100
)

# evaluate the model
model.evaluate(X_test, y_test)
```

```
Epoch 1/100
128/128 [==============================] - 1s 3ms/step - loss: 7.5484 - mae:
7.5484 - mse: 138.8465 - val_loss: 5.1579 - val_mae: 5.1579 - val_mse: 71.4915
Epoch 2/100
128/128 [==============================] - 0s 2ms/step - loss: 4.7126 - mae:
4.7126 - mse: 56.4778 - val_loss: 4.3986 - val_mae: 4.3986 - val_mse: 52.1798
Epoch 3/100
128/128 [==============================] - 0s 2ms/step - loss: 4.2758 - mae:
4.2758 - mse: 48.1720 - val_loss: 4.1968 - val_mae: 4.1968 - val_mse: 51.2747
Epoch 4/100
128/128 [==============================] - 0s 2ms/step - loss: 4.1492 - mae:
4.1492 - mse: 46.7015 - val_loss: 4.1256 - val_mae: 4.1256 - val_mse: 50.9336
Epoch 5/100
128/128 [==============================] - 0s 2ms/step - loss: 4.1178 - mae:
4.1178 - mse: 47.4057 - val_loss: 4.1225 - val_mae: 4.1225 - val_mse: 50.6077
Epoch 6/100
128/128 [==============================] - 0s 2ms/step - loss: 4.0975 - mae:
4.0975 - mse: 46.8275 - val_loss: 4.0564 - val_mae: 4.0564 - val_mse: 50.4275
Epoch 7/100
128/128 [==============================] - 0s 2ms/step - loss: 4.0366 - mae:
```

```
4.0366 - mse: 46.4801 - val_loss: 4.0687 - val_mae: 4.0687 - val_mse: 50.8511
Epoch 8/100
128/128 [==============================] - 0s 2ms/step - loss: 4.0704 - mae:
4.0704 - mse: 46.4539 - val_loss: 4.0806 - val_mae: 4.0806 - val_mse: 50.5494
Epoch 9/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9900 - mae:
3.9900 - mse: 45.4878 - val_loss: 4.0697 - val_mae: 4.0697 - val_mse: 50.5988
Epoch 10/100
128/128 [==============================] - 0s 2ms/step - loss: 4.0296 - mae:
4.0296 - mse: 46.1882 - val_loss: 4.0391 - val_mae: 4.0391 - val_mse: 50.4775
Epoch 11/100
128/128 [==============================] - 0s 3ms/step - loss: 4.0019 - mae:
4.0019 - mse: 45.7939 - val_loss: 4.0914 - val_mae: 4.0914 - val_mse: 50.0210
Epoch 12/100
128/128 [==============================] - 0s 2ms/step - loss: 4.0007 - mae:
4.0007 - mse: 45.7246 - val_loss: 4.0589 - val_mae: 4.0589 - val_mse: 50.5235
Epoch 13/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9742 - mae:
3.9742 - mse: 45.7597 - val_loss: 4.0263 - val_mae: 4.0263 - val_mse: 50.0143
Epoch 14/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9737 - mae:
3.9737 - mse: 46.2509 - val_loss: 4.0976 - val_mae: 4.0976 - val_mse: 50.4338
Epoch 15/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9359 - mae:
3.9359 - mse: 45.5384 - val_loss: 4.0963 - val_mae: 4.0963 - val_mse: 49.8453
Epoch 16/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9487 - mae:
3.9487 - mse: 45.8333 - val_loss: 4.1699 - val_mae: 4.1699 - val_mse: 50.7261
Epoch 17/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9197 - mae:
3.9197 - mse: 45.1149 - val_loss: 4.0841 - val_mae: 4.0841 - val_mse: 49.4316
Epoch 18/100
128/128 [==============================] - 0s 2ms/step - loss: 3.9163 - mae:
3.9163 - mse: 45.5167 - val_loss: 4.0543 - val_mae: 4.0543 - val_mse: 49.6576
Epoch 19/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8954 - mae:
3.8954 - mse: 45.2404 - val_loss: 4.0754 - val_mae: 4.0754 - val_mse: 49.6303
Epoch 20/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8835 - mae:
3.8835 - mse: 45.1141 - val_loss: 4.0788 - val_mae: 4.0788 - val_mse: 50.0099
Epoch 21/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8690 - mae:
3.8690 - mse: 45.0218 - val_loss: 4.0202 - val_mae: 4.0202 - val_mse: 50.4490
Epoch 22/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8870 - mae:
3.8870 - mse: 45.1902 - val_loss: 4.0560 - val_mae: 4.0560 - val_mse: 50.1610
Epoch 23/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8512 - mae:
```

```
3.8512 - mse: 44.4779 - val_loss: 4.0419 - val_mae: 4.0419 - val_mse: 50.4859
Epoch 24/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8660 - mae:
3.8660 - mse: 45.3372 - val_loss: 4.0041 - val_mae: 4.0041 - val_mse: 49.5772
Epoch 25/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8648 - mae:
3.8648 - mse: 44.6969 - val_loss: 4.0362 - val_mae: 4.0362 - val_mse: 50.2607
Epoch 26/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8463 - mae:
3.8463 - mse: 45.1143 - val_loss: 4.0037 - val_mae: 4.0037 - val_mse: 50.1196
Epoch 27/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8555 - mae:
3.8555 - mse: 44.9330 - val_loss: 4.0112 - val_mae: 4.0112 - val_mse: 49.8932
Epoch 28/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8789 - mae:
3.8789 - mse: 45.6531 - val_loss: 4.0975 - val_mae: 4.0975 - val_mse: 50.0896
Epoch 29/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8656 - mae:
3.8656 - mse: 45.3799 - val_loss: 4.0301 - val_mae: 4.0301 - val_mse: 49.5504
Epoch 30/100
128/128 [==============================] - 0s 3ms/step - loss: 3.8372 - mae:
3.8372 - mse: 44.8882 - val_loss: 4.0199 - val_mae: 4.0199 - val_mse: 49.7268
Epoch 31/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8464 - mae:
3.8464 - mse: 44.5185 - val_loss: 4.0372 - val_mae: 4.0372 - val_mse: 50.1652
Epoch 32/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8558 - mae:
3.8558 - mse: 45.2172 - val_loss: 4.0120 - val_mae: 4.0120 - val_mse: 49.9421
Epoch 33/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8323 - mae:
3.8323 - mse: 44.9111 - val_loss: 4.0073 - val_mae: 4.0073 - val_mse: 50.1507
Epoch 34/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8368 - mae:
3.8368 - mse: 44.5846 - val_loss: 4.0520 - val_mae: 4.0520 - val_mse: 50.5909
Epoch 35/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8351 - mae:
3.8351 - mse: 44.2880 - val_loss: 4.0443 - val_mae: 4.0443 - val_mse: 50.0569
Epoch 36/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8508 - mae:
3.8508 - mse: 44.8241 - val_loss: 4.0686 - val_mae: 4.0686 - val_mse: 50.3575
Epoch 37/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8730 - mae:
3.8730 - mse: 45.6932 - val_loss: 4.0077 - val_mae: 4.0077 - val_mse: 49.7425
Epoch 38/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8567 - mae:
3.8567 - mse: 45.3104 - val_loss: 4.0024 - val_mae: 4.0024 - val_mse: 49.5503
Epoch 39/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8123 - mae:
```

```
3.8123 - mse: 44.0819 - val_loss: 4.0273 - val_mae: 4.0273 - val_mse: 50.6720
Epoch 40/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8031 - mae:
3.8031 - mse: 44.2044 - val_loss: 4.0177 - val_mae: 4.0177 - val_mse: 50.3299
Epoch 41/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8178 - mae:
3.8178 - mse: 44.4506 - val_loss: 3.9938 - val_mae: 3.9938 - val_mse: 49.4292
Epoch 42/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7931 - mae:
3.7931 - mse: 43.7464 - val_loss: 3.9935 - val_mae: 3.9935 - val_mse: 49.6499
Epoch 43/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8124 - mae:
3.8124 - mse: 44.5085 - val_loss: 4.0283 - val_mae: 4.0283 - val_mse: 50.0198
Epoch 44/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7705 - mae:
3.7705 - mse: 43.2404 - val_loss: 3.9769 - val_mae: 3.9769 - val_mse: 49.1438
Epoch 45/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8157 - mae:
3.8157 - mse: 44.2807 - val_loss: 3.9914 - val_mae: 3.9914 - val_mse: 49.9685
Epoch 46/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8044 - mae:
3.8044 - mse: 43.9938 - val_loss: 4.0042 - val_mae: 4.0042 - val_mse: 49.5597
Epoch 47/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8095 - mae:
3.8095 - mse: 43.8789 - val_loss: 3.9825 - val_mae: 3.9825 - val_mse: 49.7848
Epoch 48/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7829 - mae:
3.7829 - mse: 43.7901 - val_loss: 3.9996 - val_mae: 3.9996 - val_mse: 49.3999
Epoch 49/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8086 - mae:
3.8086 - mse: 43.7681 - val_loss: 3.9914 - val_mae: 3.9914 - val_mse: 50.0081
Epoch 50/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8159 - mae:
3.8159 - mse: 44.2112 - val_loss: 4.0107 - val_mae: 4.0107 - val_mse: 49.5435
Epoch 51/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7878 - mae:
3.7878 - mse: 43.4658 - val_loss: 4.0032 - val_mae: 4.0032 - val_mse: 49.6451
Epoch 52/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8143 - mae:
3.8143 - mse: 44.5602 - val_loss: 3.9996 - val_mae: 3.9996 - val_mse: 48.9782
Epoch 53/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7929 - mae:
3.7929 - mse: 43.4312 - val_loss: 3.9751 - val_mae: 3.9751 - val_mse: 48.5395
Epoch 54/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8136 - mae:
3.8136 - mse: 43.9847 - val_loss: 3.9749 - val_mae: 3.9749 - val_mse: 48.8883
Epoch 55/100
128/128 [==============================] - 0s 3ms/step - loss: 3.8028 - mae:
```

```
3.8028 - mse: 43.4940 - val_loss: 3.9804 - val_mae: 3.9804 - val_mse: 49.1514
Epoch 56/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7769 - mae:
3.7769 - mse: 43.6930 - val_loss: 3.9907 - val_mae: 3.9907 - val_mse: 49.4545
Epoch 57/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8154 - mae:
3.8154 - mse: 44.3153 - val_loss: 3.9850 - val_mae: 3.9850 - val_mse: 49.4492
Epoch 58/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7905 - mae:
3.7905 - mse: 43.8287 - val_loss: 3.9906 - val_mae: 3.9906 - val_mse: 48.4052
Epoch 59/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7868 - mae:
3.7868 - mse: 43.7335 - val_loss: 3.9746 - val_mae: 3.9746 - val_mse: 48.8971
Epoch 60/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7786 - mae:
3.7786 - mse: 43.2525 - val_loss: 3.9837 - val_mae: 3.9837 - val_mse: 49.2148
Epoch 61/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7970 - mae:
3.7970 - mse: 43.6412 - val_loss: 4.0032 - val_mae: 4.0032 - val_mse: 49.3306
Epoch 62/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7746 - mae:
3.7746 - mse: 43.2427 - val_loss: 3.9891 - val_mae: 3.9891 - val_mse: 49.0809
Epoch 63/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7771 - mae:
3.7771 - mse: 43.4581 - val_loss: 3.9907 - val_mae: 3.9907 - val_mse: 48.3357
Epoch 64/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8115 - mae:
3.8115 - mse: 44.0086 - val_loss: 3.9773 - val_mae: 3.9773 - val_mse: 49.2669
Epoch 65/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7662 - mae:
3.7662 - mse: 43.3936 - val_loss: 3.9802 - val_mae: 3.9802 - val_mse: 49.4459
Epoch 66/100
128/128 [==============================] - 0s 2ms/step - loss: 3.8030 - mae:
3.8030 - mse: 44.5192 - val_loss: 3.9757 - val_mae: 3.9757 - val_mse: 49.0841
Epoch 67/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7893 - mae:
3.7893 - mse: 44.0116 - val_loss: 4.0152 - val_mae: 4.0152 - val_mse: 48.9692
Epoch 68/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7862 - mae:
3.7862 - mse: 43.3637 - val_loss: 3.9850 - val_mae: 3.9850 - val_mse: 49.4381
Epoch 69/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7798 - mae:
3.7798 - mse: 43.5573 - val_loss: 3.9791 - val_mae: 3.9791 - val_mse: 48.7496
Epoch 70/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7940 - mae:
3.7940 - mse: 43.4163 - val_loss: 3.9793 - val_mae: 3.9793 - val_mse: 48.7267
Epoch 71/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7702 - mae:
```

```
3.7702 - mse: 42.9481 - val_loss: 3.9950 - val_mae: 3.9950 - val_mse: 49.7794
Epoch 72/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7753 - mae:
3.7753 - mse: 43.8113 - val_loss: 4.0153 - val_mae: 4.0153 - val_mse: 50.1511
Epoch 73/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7875 - mae:
3.7875 - mse: 43.7855 - val_loss: 4.0062 - val_mae: 4.0062 - val_mse: 49.2320
Epoch 74/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7686 - mae:
3.7686 - mse: 43.0304 - val_loss: 3.9786 - val_mae: 3.9786 - val_mse: 48.8228
Epoch 75/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7593 - mae:
3.7593 - mse: 42.8296 - val_loss: 4.0067 - val_mae: 4.0067 - val_mse: 49.6042
Epoch 76/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7905 - mae:
3.7905 - mse: 43.7053 - val_loss: 4.0067 - val_mae: 4.0067 - val_mse: 50.0978
Epoch 77/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7956 - mae:
3.7956 - mse: 43.3893 - val_loss: 4.0370 - val_mae: 4.0370 - val_mse: 50.1389
Epoch 78/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7600 - mae:
3.7600 - mse: 43.4185 - val_loss: 3.9857 - val_mae: 3.9857 - val_mse: 49.1601
Epoch 79/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7702 - mae:
3.7702 - mse: 43.0458 - val_loss: 3.9714 - val_mae: 3.9714 - val_mse: 48.6256
Epoch 80/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7658 - mae:
3.7658 - mse: 43.2500 - val_loss: 3.9881 - val_mae: 3.9881 - val_mse: 49.2811
Epoch 81/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7533 - mae:
3.7533 - mse: 42.8163 - val_loss: 3.9904 - val_mae: 3.9904 - val_mse: 49.4579
Epoch 82/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7378 - mae:
3.7378 - mse: 43.1784 - val_loss: 4.0200 - val_mae: 4.0200 - val_mse: 49.1161
Epoch 83/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7701 - mae:
3.7701 - mse: 43.3063 - val_loss: 3.9872 - val_mae: 3.9872 - val_mse: 49.2098
Epoch 84/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7286 - mae:
3.7286 - mse: 42.4433 - val_loss: 3.9843 - val_mae: 3.9843 - val_mse: 48.7519
Epoch 85/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7403 - mae:
3.7403 - mse: 42.9612 - val_loss: 4.0599 - val_mae: 4.0599 - val_mse: 49.4860
Epoch 86/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7653 - mae:
3.7653 - mse: 43.3011 - val_loss: 4.0026 - val_mae: 4.0026 - val_mse: 48.6884
Epoch 87/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7677 - mae:
```

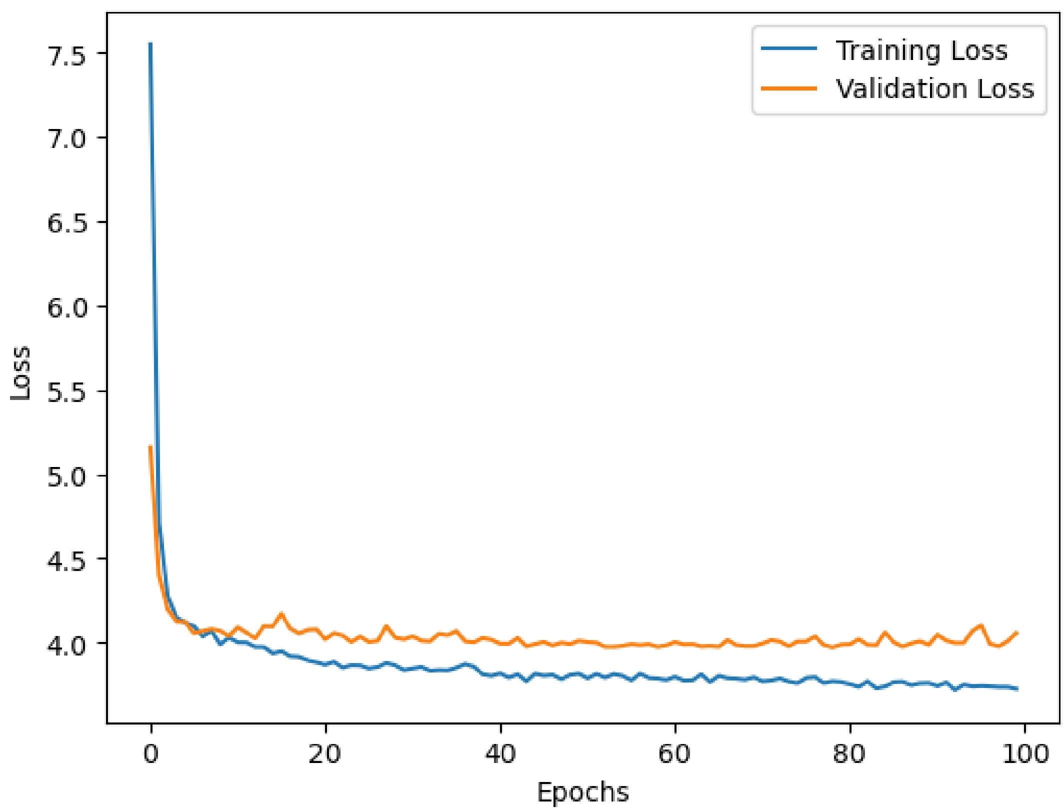
```
3.7677 - mse: 43.0881 - val_loss: 3.9752 - val_mae: 3.9752 - val_mse: 48.5150
Epoch 88/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7478 - mae:
3.7478 - mse: 42.5929 - val_loss: 3.9956 - val_mae: 3.9956 - val_mse: 49.5601
Epoch 89/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7586 - mae:
3.7586 - mse: 42.7496 - val_loss: 4.0081 - val_mae: 4.0081 - val_mse: 49.7821
Epoch 90/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7603 - mae:
3.7603 - mse: 42.9358 - val_loss: 3.9867 - val_mae: 3.9867 - val_mse: 49.6547
Epoch 91/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7425 - mae:
3.7425 - mse: 43.0991 - val_loss: 4.0476 - val_mae: 4.0476 - val_mse: 49.5374
Epoch 92/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7638 - mae:
3.7638 - mse: 42.9811 - val_loss: 4.0124 - val_mae: 4.0124 - val_mse: 48.9657
Epoch 93/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7197 - mae:
3.7197 - mse: 42.0719 - val_loss: 3.9982 - val_mae: 3.9982 - val_mse: 49.2365
Epoch 94/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7513 - mae:
3.7513 - mse: 42.3687 - val_loss: 3.9987 - val_mae: 3.9987 - val_mse: 49.3001
Epoch 95/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7401 - mae:
3.7401 - mse: 42.3538 - val_loss: 4.0696 - val_mae: 4.0696 - val_mse: 51.4032
Epoch 96/100
128/128 [==============================] - 0s 3ms/step - loss: 3.7432 - mae:
3.7432 - mse: 42.8678 - val_loss: 4.1016 - val_mae: 4.1016 - val_mse: 51.4860
Epoch 97/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7406 - mae:
3.7406 - mse: 42.7854 - val_loss: 3.9927 - val_mae: 3.9927 - val_mse: 48.7736
Epoch 98/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7370 - mae:
3.7370 - mse: 42.4465 - val_loss: 3.9769 - val_mae: 3.9769 - val_mse: 48.7225
Epoch 99/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7370 - mae:
3.7370 - mse: 42.3312 - val_loss: 4.0085 - val_mae: 4.0085 - val_mse: 50.1510
Epoch 100/100
128/128 [==============================] - 0s 2ms/step - loss: 3.7261 - mae:
3.7261 - mse: 42.2215 - val_loss: 4.0560 - val_mae: 4.0560 - val_mse: 51.0932
32/32 [==============================] - 0s 1ms/step - loss: 4.0560 - mae:
4.0560 - mse: 51.0932
```

[74]:
```
[4.055957794189453, 4.055957794189453, 51.09320831298828]
```

[75]:
```
import matplotlib.pyplot as plt
```

```python
# Evaluating model performance
mse = model.evaluate(X_test, y_test)
rmse = np.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Plotting training/validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
32/32 [==============================] - 0s 1ms/step - loss: 4.0560 - mae:
4.0560 - mse: 51.0932
Mean Squared Error (MSE): [4.055957794189453, 4.055957794189453,
51.09320831298828]
Root Mean Squared Error (RMSE): [2.01394086 2.01394086 7.14795134]
```



2. Optimization Techniques (Using Different Optimizers and Learning Rate Scheduling)

[76]:
```
# make a NN
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# ! adjust the layers as need be
model = keras.Sequential([
    layers.Dense(8, activation='relu', input_shape=[12]),  # Update input shape␣
 ↪to match your feature count
    layers.Dropout(0.2),  # Adding dropout to the first hidden layer
    layers.Dense(12, activation='relu'),
    layers.Dense(1)
])


# Example of using a different optimizer (RMSprop) and adding learning rate␣
 ↪scheduling
opt = keras.optimizers.RMSprop(learning_rate=0.001)
model.compile(optimizer=opt, loss='mae', metrics=['mae', 'mse'])

# Learning Rate Scheduling (ReduceLROnPlateau callback)
lr_scheduler = keras.callbacks.ReduceLROnPlateau(factor=0.5, patience=3,␣
 ↪min_lr=0.0001)
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪callbacks=[lr_scheduler])

# evaluate the model
model.evaluate(X_test, y_test)
```

```
128/128 [==============================] - 1s 3ms/step - loss: 7.0867 - mae:
7.0867 - mse: 130.7226 - val_loss: 5.0770 - val_mae: 5.0770 - val_mse: 73.8859 -
lr: 0.0010
32/32 [==============================] - 0s 1ms/step - loss: 5.0770 - mae:
5.0770 - mse: 73.8859
```

[76]:
```
[5.076963901519775, 5.076963901519775, 73.88587188720703]
```

[149]:
```
# Evaluating model performance
mse = model.evaluate(X_test, y_test)
rmse = np.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
32/32 [==============================] - 0s 3ms/step - loss: 3.9619 - mae:
3.9619 - mse: 49.5770
Mean Squared Error (MSE): [3.9618654251098633, 3.9618654251098633,
49.57695770263672]
```

```
Root Mean Squared Error (RMSE): [1.99044352 1.99044352 7.04109066]
```

3. Early Stopping (Using EarlyStopping Callback)

[89]:
```python
# make a NN
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# ! adjust the layers as need be
model = keras.Sequential([
    layers.Dense(8, activation='relu', input_shape=[12]),  # Update input shape␣
↪to match your feature count
    layers.Dropout(0.2),  # Adding dropout to the first hidden layer
    layers.Dense(12, activation='relu'),
    layers.Dense(1)
])


# Example of using a different optimizer (RMSprop) and adding learning rate␣
↪scheduling
opt = keras.optimizers.RMSprop(learning_rate=0.001)
model.compile(optimizer=opt, loss='mae', metrics=['mae', 'mse'])

# Early Stopping with patience set to stop training when the validation loss␣
↪stops improving
early_stopping = keras.callbacks.EarlyStopping(patience=5,␣
↪restore_best_weights=True)
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪callbacks=[early_stopping])


# evaluate the model
model.evaluate(X_test, y_test)
```

```
128/128 [==============================] - 1s 4ms/step - loss: 6.9221 - mae:
6.9221 - mse: 125.6306 - val_loss: 5.1018 - val_mae: 5.1018 - val_mse: 66.9033
32/32 [==============================] - 0s 2ms/step - loss: 5.1018 - mae:
5.1018 - mse: 66.9033
```

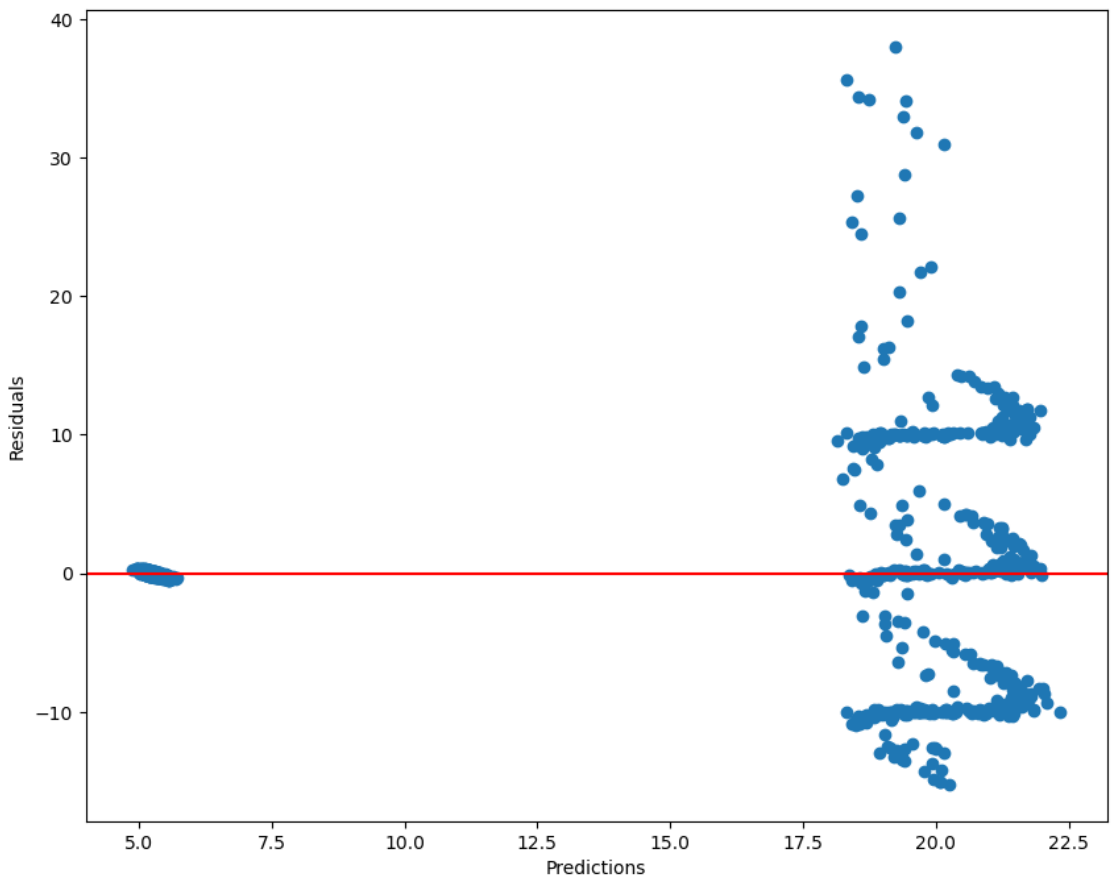
[89]:
```
[5.101824760437012, 5.101824760437012, 66.90327453613281]
```

[148]:
```python
# Evaluating model performance
mse = model.evaluate(X_test, y_test)
rmse = np.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
32/32 [==============================] - 0s 6ms/step - loss: 3.9619 - mae:
3.9619 - mse: 49.5770
Mean Squared Error (MSE): [3.9618654251098633, 3.9618654251098633,
49.57695770263672]
Root Mean Squared Error (RMSE): [1.99044352 1.99044352 7.04109066]
```

1.4 Explore my best model

[150]:
```python
# Because my final_svm_model is my best, I will plot residuals for it
# Plotting residuals
plt.figure(figsize=(10, 8))
plt.scatter(x=svm_predictions, y=y_test - svm_predictions)
plt.xlabel('Predictions')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='-')
plt.show()
```



1.5 Predict

Based on the above, the final_SVM_model is my best model, I will use it to make my predictions

[130]:

```python
# import and display the test_2.csv file
test_df = pd.read_csv('test_2.csv')

# print the df shape
print(test_df.shape)

# print the df head
test_df.head()
```

```
(1260, 13)
```

[130]:

```
   Unnamed: 0 shipment_id       send_timestamp pick_up_point drop_off_point  \
0           0     S002736  2019-10-04 14:27:04             A              Y
1           1     S002738  2020-01-07 09:39:35             A              Y
2           2     S005739  2020-04-11 11:58:10             A              Y
3           3     S008722  2019-06-23 11:54:41             A              Y
4           4     S009737  2019-11-20 20:18:01             A              Y

  source_country destination_country  freight_cost  gross_weight  \
0             GB                  IN         86.81         100.0
1             GB                  IN         94.43        1006.0
2             GB                  IN         93.55         321.0
3             GB                  IN         88.74         355.0
4             GB                  IN         92.83         115.0

   shipment_charges shipment_mode shipping_company selected
0              0.75           Air              SC3        Y
1              0.75           Air              SC3        Y
2              1.05           Air              SC2        Y
3              1.05           Air              SC2        Y
4              1.05           Air              SC2        Y
```

[131]:

```python
# make the changes to the test_df that were made to the training one
# drop the following columns: shipment_id, pickup_point, source_country,␣
↪selected because they only have 1 value
test_df.drop(['Unnamed: 0', 'shipment_id', 'pick_up_point', 'source_country',␣
↪'selected'], axis=1, inplace=True)

# # Encode the following categorical variables into numeric ones:␣
↪drop_off_point, destination_country, shipment_mode, shipping_company
colums_to_encode = ['drop_off_point', 'destination_country', 'shipment_mode',␣
↪'shipping_company']

# Encode 'travel_from' and 'car_type' columns
for column in colums_to_encode:
    test_df[column + '_encoded'] = label_encoder.fit_transform(test_df[column])
    # drop the column
```

```
    test_df.drop(column, axis=1, inplace=True)

# get the day of year column from the send_timestamp column
test_df['send_timestamp'] = pd.to_datetime(test_df['send_timestamp'])  #␣
 ↪Convert to datetime if not already in datetime format

# get the day of year column from the send_timestamp column
test_df['day_of_year_sent'] = test_df['send_timestamp'].dt.dayofyear

# Extracting hour, minute, second
test_df['hour'] = test_df['send_timestamp'].dt.hour
test_df['minute'] = test_df['send_timestamp'].dt.minute
test_df['second'] = test_df['send_timestamp'].dt.second

# Applying cyclical encoding (sine and cosine transformations) for cyclical␣
 ↪patterns
test_df['hour_sin'] = np.sin(2 * np.pi * test_df['hour'] / 24.0)
# test_df['hour_cos'] = np.cos(2 * np.pi * test_df['hour'] / 24.0)

test_df['minute_sin'] = np.sin(2 * np.pi * test_df['minute'] / 60.0)
test_df['minute_cos'] = np.cos(2 * np.pi * test_df['minute'] / 60.0)

test_df['second_sin'] = np.sin(2 * np.pi * test_df['second'] / 60.0)
test_df['second_cos'] = np.cos(2 * np.pi * test_df['second'] / 60.0)

# Drop the original 'send_timestamp' column, as well as the hour, minute,␣
 ↪second columns
test_df.drop('send_timestamp', axis=1, inplace=True)
test_df.drop('hour', axis=1, inplace=True)
test_df.drop('minute', axis=1, inplace=True)
test_df.drop('second', axis=1, inplace=True)

# drop drop off point encoded
test_df.drop('drop_off_point_encoded', axis=1, inplace=True)

# scale the gross_weight column and the freight_cost column and␣
 ↪send_day_of_year column

test_df['gross_weight'] = scaler.fit_transform(test_df[['gross_weight']])
test_df['freight_cost'] = scaler.fit_transform(test_df[['freight_cost']])
test_df['day_of_year_sent'] = scaler.
 ↪fit_transform(test_df[['day_of_year_sent']])

# print the df shape
print(test_df.shape)

# print the df head
```

```
test_df.head()
```

(1260, 12)

[131]:

```
   freight_cost  gross_weight  shipment_charges  destination_country_encoded  \
0     -0.857283     -0.655580              0.75                            1
1      0.643164      0.016892              0.75                            1
2      0.469884     -0.491544              1.05                            1
3     -0.477248     -0.466308              1.05                            1
4      0.328110     -0.644446              1.05                            1

   shipment_mode_encoded  shipping_company_encoded  day_of_year_sent  \
0                      0                         2          0.950624
1                      0                         2         -1.697783
2                      0                         1         -0.765936
3                      0                         1         -0.059694
4                      0                         1          1.411643

   hour_sin  minute_sin  minute_cos  second_sin  second_cos
0 -0.500000    0.309017   -0.951057    0.406737    0.913545
1  0.707107   -0.809017   -0.587785   -0.500000   -0.866025
2  0.258819   -0.207912    0.978148    0.866025    0.500000
3  0.258819   -0.587785    0.809017   -0.913545   -0.406737
4 -0.866025    0.951057   -0.309017    0.104528    0.994522
```

[132]:

```
# with my test_df as my X, predict the shipping time
test_predictions = final_svm_model.predict(test_df)

# print the predictions
print(test_predictions)
```

```
[ 5.1983236   5.22173045  5.18848975 … 19.72545797 18.9379961
 19.15871632]
```

[134]:

```
# make a df out of the initial test_df's shipment_id column and the predictions
test_predictions_df = pd.DataFrame(pd.read_csv('test_2.csv')['shipment_id'])
test_predictions_df['shipping_time'] = test_predictions

# print the df shape
print(test_predictions_df.shape)

# print the df head
test_predictions_df.head()
```

(1260, 2)

[134]:

	shipment_id	shipping_time
0	S002736	5.198324
1	S002738	5.221730
2	S005739	5.188490
3	S008722	5.151506
4	S009737	5.420550

[135]:
```
# save the df as a csv file called submission.csv
test_predictions_df.to_csv('submission.csv', index=False)
```

1.5.1 Out Of Challenge and Coursework Scope

I will attempt other models for increased performance 1. Multi-linear regression 2. Decision tree 3. Random Forest 4. XGBoost, LightBoost and CatBoost 5. Lasso regression model 6. KNN model 7. Gaussian model

[156]:
```
# make a multiple linear regression model
from sklearn.linear_model import LinearRegression

# Initialize the model
linear_model = LinearRegression()

# Train the model
linear_model.fit(X_train, y_train)

# Predict using the model
linear_predictions = linear_model.predict(X_test)

# Evaluating model performance
linear_mse = mean_squared_error(y_test, linear_predictions)
linear_rmse = np.sqrt(linear_mse)
linear_mae = mean_absolute_error(y_test, linear_predictions)

print(f"Linear Regression Mean Squared Error (MSE): {linear_mse}")
print(f"Linear Regression Root Mean Squared Error (RMSE): {linear_rmse}")
print(f"Linear Regression Mean Absolute Error (MAE): {linear_mae}")
```

```
Linear Regression Mean Squared Error (MSE): 47.410224058359006
Linear Regression Root Mean Squared Error (RMSE): 6.885508264344689
Linear Regression Mean Absolute Error (MAE): 4.215050984704281
```

[137]:
```
# make a decision tree model
from sklearn.tree import DecisionTreeRegressor

# Initialize the model
tree_model = DecisionTreeRegressor()

# Train the model
```

```
tree_model.fit(X_train, y_train)

# Predict using the model
tree_predictions = tree_model.predict(X_test)

# Evaluating model performance
tree_mse = mean_squared_error(y_test, tree_predictions)
tree_rmse = np.sqrt(tree_mse)

print(f"Decision Tree Mean Squared Error (MSE): {tree_mse}")
print(f"Decision Tree Root Mean Squared Error (RMSE): {tree_rmse}")
```

```
Decision Tree Mean Squared Error (MSE): 93.26478625735132
Decision Tree Root Mean Squared Error (RMSE): 9.657369530951549
```

[138]:
```
# make a random forest model
from sklearn.ensemble import RandomForestRegressor

# Initialize the model
forest_model = RandomForestRegressor()

# Train the model
forest_model.fit(X_train, y_train)

# Predict using the model
forest_predictions = forest_model.predict(X_test)

# Evaluating model performance
forest_mse = mean_squared_error(y_test, forest_predictions)
forest_rmse = np.sqrt(forest_mse)

print(f"Random Forest Mean Squared Error (MSE): {forest_mse}")
print(f"Random Forest Root Mean Squared Error (RMSE): {forest_rmse}")
```

```
Random Forest Mean Squared Error (MSE): 48.26718703353946
Random Forest Root Mean Squared Error (RMSE): 6.9474590343189115
```

[160]:
```
# make XGBoost, LightBoost and CatBoost models
# !pip install xgboost, lightgbm, catboost
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor

# Initialize the models
xgb_model = XGBRegressor()
lgb_model = LGBMRegressor()
cat_model = CatBoostRegressor()
```

```python
# Train the models
xgb_model.fit(X_train, y_train)
lgb_model.fit(X_train, y_train)
cat_model.fit(X_train, y_train)

# Predict using the models
xgb_predictions = xgb_model.predict(X_test)
lgb_predictions = lgb_model.predict(X_test)
cat_predictions = cat_model.predict(X_test)

# Evaluating model performance
xgb_mse = mean_squared_error(y_test, xgb_predictions)
xgb_rmse = np.sqrt(xgb_mse)

lgb_mse = mean_squared_error(y_test, lgb_predictions)
lgb_rmse = np.sqrt(lgb_mse)

cat_mse = mean_squared_error(y_test, cat_predictions)
cat_rmse = np.sqrt(cat_mse)

print(f"XGBoost Mean Squared Error (MSE): {xgb_mse}")
print(f"XGBoost Root Mean Squared Error (RMSE): {xgb_rmse}")

print(f"LightBoost Mean Squared Error (MSE): {lgb_mse}")
print(f"LightBoost Root Mean Squared Error (RMSE): {lgb_rmse}")

print(f"CatBoost Mean Squared Error (MSE): {cat_mse}")
print(f"CatBoost Root Mean Squared Error (RMSE): {cat_rmse}")
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.019407 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 973
[LightGBM] [Info] Number of data points in the train set: 4091, number of used
features: 12
[LightGBM] [Info] Start training from score 12.608936
Learning rate set to 0.05115
0:      learn: 9.9281883        total: 48.9ms   remaining: 48.9s
1:      learn: 9.6489983        total: 50.1ms   remaining: 25s
2:      learn: 9.3964511        total: 51.1ms   remaining: 17s
3:      learn: 9.1637078        total: 52.5ms   remaining: 13.1s
4:      learn: 8.9461965        total: 53.6ms   remaining: 10.7s
5:      learn: 8.7391259        total: 54.6ms   remaining: 9.05s
6:      learn: 8.5478672        total: 55.5ms   remaining: 7.88s
7:      learn: 8.3769073        total: 56.2ms   remaining: 6.97s
8:      learn: 8.2110793        total: 57.3ms   remaining: 6.31s
```

```
9:      learn: 8.0652199        total: 58.2ms   remaining: 5.76s
10:     learn: 7.9299116        total: 59.1ms   remaining: 5.31s
11:     learn: 7.8072758        total: 59.9ms   remaining: 4.93s
12:     learn: 7.6931544        total: 60.9ms   remaining: 4.63s
13:     learn: 7.5843649        total: 62ms     remaining: 4.37s
14:     learn: 7.4905612        total: 62.6ms   remaining: 4.11s
15:     learn: 7.4052778        total: 63.2ms   remaining: 3.88s
16:     learn: 7.3204380        total: 64.2ms   remaining: 3.71s
17:     learn: 7.2425391        total: 65.1ms   remaining: 3.55s
18:     learn: 7.1779815        total: 65.8ms   remaining: 3.4s
19:     learn: 7.1124557        total: 66.7ms   remaining: 3.27s
20:     learn: 7.0515262        total: 67.8ms   remaining: 3.16s
21:     learn: 6.9987410        total: 69.2ms   remaining: 3.08s
22:     learn: 6.9468998        total: 70.3ms   remaining: 2.98s
23:     learn: 6.9021667        total: 71.2ms   remaining: 2.9s
24:     learn: 6.8603020        total: 72.3ms   remaining: 2.82s
25:     learn: 6.8237295        total: 73.2ms   remaining: 2.74s
26:     learn: 6.7892523        total: 74.2ms   remaining: 2.67s
27:     learn: 6.7578345        total: 75.5ms   remaining: 2.62s
28:     learn: 6.7242912        total: 76.8ms   remaining: 2.57s
29:     learn: 6.6970637        total: 78.2ms   remaining: 2.53s
30:     learn: 6.6716797        total: 79.6ms   remaining: 2.49s
31:     learn: 6.6486082        total: 80.6ms   remaining: 2.44s
32:     learn: 6.6274706        total: 81.7ms   remaining: 2.39s
33:     learn: 6.6017937        total: 83.2ms   remaining: 2.36s
34:     learn: 6.5815532        total: 84.4ms   remaining: 2.33s
35:     learn: 6.5622236        total: 85.4ms   remaining: 2.29s
36:     learn: 6.5442042        total: 86.4ms   remaining: 2.25s
37:     learn: 6.5331317        total: 87.4ms   remaining: 2.21s
38:     learn: 6.5203564        total: 88.5ms   remaining: 2.18s
39:     learn: 6.5022032        total: 89.5ms   remaining: 2.15s
40:     learn: 6.4846069        total: 90.5ms   remaining: 2.12s
41:     learn: 6.4699094        total: 91.6ms   remaining: 2.09s
42:     learn: 6.4583693        total: 92.9ms   remaining: 2.07s
43:     learn: 6.4527511        total: 93.5ms   remaining: 2.03s
44:     learn: 6.4413601        total: 95.2ms   remaining: 2.02s
45:     learn: 6.4285436        total: 97.2ms   remaining: 2.02s
46:     learn: 6.4187841        total: 98.4ms   remaining: 2s
47:     learn: 6.4019171        total: 100ms    remaining: 1.99s
48:     learn: 6.3943185        total: 102ms    remaining: 1.98s
49:     learn: 6.3868821        total: 103ms    remaining: 1.96s
50:     learn: 6.3753339        total: 104ms    remaining: 1.94s
51:     learn: 6.3627411        total: 106ms    remaining: 1.93s
52:     learn: 6.3582881        total: 109ms    remaining: 1.95s
53:     learn: 6.3504446        total: 110ms    remaining: 1.93s
54:     learn: 6.3440683        total: 111ms    remaining: 1.91s
55:     learn: 6.3402498        total: 112ms    remaining: 1.89s
56:     learn: 6.3323970        total: 114ms    remaining: 1.88s
```

```
57:     learn: 6.3294481        total: 115ms    remaining: 1.87s
58:     learn: 6.3219682        total: 116ms    remaining: 1.85s
59:     learn: 6.3136669        total: 118ms    remaining: 1.84s
60:     learn: 6.3065501        total: 120ms    remaining: 1.84s
61:     learn: 6.2934727        total: 124ms    remaining: 1.88s
62:     learn: 6.2879228        total: 128ms    remaining: 1.9s
63:     learn: 6.2748640        total: 134ms    remaining: 1.96s
64:     learn: 6.2715716        total: 137ms    remaining: 1.96s
65:     learn: 6.2625541        total: 142ms    remaining: 2.01s
66:     learn: 6.2599361        total: 144ms    remaining: 2.01s
67:     learn: 6.2560140        total: 147ms    remaining: 2.01s
68:     learn: 6.2501498        total: 154ms    remaining: 2.08s
69:     learn: 6.2432030        total: 157ms    remaining: 2.09s
70:     learn: 6.2389950        total: 159ms    remaining: 2.07s
71:     learn: 6.2352193        total: 161ms    remaining: 2.08s
72:     learn: 6.2296374        total: 164ms    remaining: 2.08s
73:     learn: 6.2255024        total: 165ms    remaining: 2.07s
74:     learn: 6.2169692        total: 167ms    remaining: 2.06s
75:     learn: 6.2115674        total: 169ms    remaining: 2.06s
76:     learn: 6.2052835        total: 171ms    remaining: 2.05s
77:     learn: 6.1998750        total: 174ms    remaining: 2.05s
78:     learn: 6.1895973        total: 178ms    remaining: 2.07s
79:     learn: 6.1846339        total: 181ms    remaining: 2.08s
80:     learn: 6.1801320        total: 183ms    remaining: 2.07s
81:     learn: 6.1773939        total: 184ms    remaining: 2.06s
82:     learn: 6.1739498        total: 186ms    remaining: 2.06s
83:     learn: 6.1695097        total: 188ms    remaining: 2.05s
84:     learn: 6.1665363        total: 190ms    remaining: 2.05s
85:     learn: 6.1584343        total: 192ms    remaining: 2.04s
86:     learn: 6.1553258        total: 194ms    remaining: 2.03s
87:     learn: 6.1501281        total: 195ms    remaining: 2.02s
88:     learn: 6.1484625        total: 196ms    remaining: 2.01s
89:     learn: 6.1443215        total: 197ms    remaining: 2s
90:     learn: 6.1397420        total: 199ms    remaining: 1.99s
91:     learn: 6.1359086        total: 200ms    remaining: 1.98s
92:     learn: 6.1315079        total: 202ms    remaining: 1.97s
93:     learn: 6.1230767        total: 203ms    remaining: 1.96s
94:     learn: 6.1213316        total: 206ms    remaining: 1.96s
95:     learn: 6.1179027        total: 208ms    remaining: 1.96s
96:     learn: 6.1139072        total: 209ms    remaining: 1.95s
97:     learn: 6.1111035        total: 211ms    remaining: 1.94s
98:     learn: 6.1059778        total: 212ms    remaining: 1.93s
99:     learn: 6.1006183        total: 213ms    remaining: 1.92s
100:    learn: 6.0972782        total: 215ms    remaining: 1.92s
101:    learn: 6.0933957        total: 217ms    remaining: 1.91s
102:    learn: 6.0836287        total: 218ms    remaining: 1.9s
103:    learn: 6.0786945        total: 219ms    remaining: 1.89s
104:    learn: 6.0740403        total: 221ms    remaining: 1.88s
```

```
105:    learn: 6.0703248        total: 222ms    remaining: 1.87s
106:    learn: 6.0640158        total: 223ms    remaining: 1.86s
107:    learn: 6.0613213        total: 225ms    remaining: 1.85s
108:    learn: 6.0574320        total: 226ms    remaining: 1.84s
109:    learn: 6.0507908        total: 227ms    remaining: 1.83s
110:    learn: 6.0487528        total: 228ms    remaining: 1.82s
111:    learn: 6.0415388        total: 229ms    remaining: 1.82s
112:    learn: 6.0322097        total: 230ms    remaining: 1.81s
113:    learn: 6.0280369        total: 232ms    remaining: 1.8s
114:    learn: 6.0251334        total: 233ms    remaining: 1.79s
115:    learn: 6.0203764        total: 234ms    remaining: 1.78s
116:    learn: 6.0182684        total: 235ms    remaining: 1.77s
117:    learn: 6.0159018        total: 237ms    remaining: 1.77s
118:    learn: 6.0084514        total: 238ms    remaining: 1.76s
119:    learn: 6.0035935        total: 239ms    remaining: 1.75s
120:    learn: 6.0006435        total: 240ms    remaining: 1.75s
121:    learn: 5.9973035        total: 241ms    remaining: 1.74s
122:    learn: 5.9898547        total: 243ms    remaining: 1.73s
123:    learn: 5.9845469        total: 244ms    remaining: 1.72s
124:    learn: 5.9814146        total: 245ms    remaining: 1.71s
125:    learn: 5.9770103        total: 246ms    remaining: 1.71s
126:    learn: 5.9642430        total: 247ms    remaining: 1.7s
127:    learn: 5.9605461        total: 249ms    remaining: 1.7s
128:    learn: 5.9565301        total: 250ms    remaining: 1.69s
129:    learn: 5.9537847        total: 252ms    remaining: 1.69s
130:    learn: 5.9486238        total: 253ms    remaining: 1.68s
131:    learn: 5.9417065        total: 254ms    remaining: 1.67s
132:    learn: 5.9377178        total: 255ms    remaining: 1.66s
133:    learn: 5.9340709        total: 256ms    remaining: 1.66s
134:    learn: 5.9300630        total: 257ms    remaining: 1.65s
135:    learn: 5.9253957        total: 258ms    remaining: 1.64s
136:    learn: 5.9221263        total: 260ms    remaining: 1.63s
137:    learn: 5.9196418        total: 261ms    remaining: 1.63s
138:    learn: 5.9160846        total: 262ms    remaining: 1.62s
139:    learn: 5.9111457        total: 263ms    remaining: 1.62s
140:    learn: 5.9076426        total: 264ms    remaining: 1.61s
141:    learn: 5.9011659        total: 265ms    remaining: 1.6s
142:    learn: 5.8963335        total: 267ms    remaining: 1.6s
143:    learn: 5.8930393        total: 268ms    remaining: 1.59s
144:    learn: 5.8895428        total: 269ms    remaining: 1.59s
145:    learn: 5.8858699        total: 270ms    remaining: 1.58s
146:    learn: 5.8808692        total: 271ms    remaining: 1.57s
147:    learn: 5.8770256        total: 272ms    remaining: 1.57s
148:    learn: 5.8743908        total: 273ms    remaining: 1.56s
149:    learn: 5.8706803        total: 275ms    remaining: 1.56s
150:    learn: 5.8657738        total: 276ms    remaining: 1.55s
151:    learn: 5.8614520        total: 277ms    remaining: 1.55s
152:    learn: 5.8552376        total: 279ms    remaining: 1.54s
```

```
153:	learn: 5.8531471	total: 280ms	remaining: 1.54s
154:	learn: 5.8474501	total: 281ms	remaining: 1.53s
155:	learn: 5.8422393	total: 282ms	remaining: 1.53s
156:	learn: 5.8388052	total: 283ms	remaining: 1.52s
157:	learn: 5.8331295	total: 284ms	remaining: 1.51s
158:	learn: 5.8276319	total: 285ms	remaining: 1.51s
159:	learn: 5.8238676	total: 286ms	remaining: 1.5s
160:	learn: 5.8213518	total: 288ms	remaining: 1.5s
161:	learn: 5.8164156	total: 289ms	remaining: 1.49s
162:	learn: 5.8140934	total: 291ms	remaining: 1.49s
163:	learn: 5.8106916	total: 293ms	remaining: 1.49s
164:	learn: 5.8075528	total: 294ms	remaining: 1.49s
165:	learn: 5.8059330	total: 295ms	remaining: 1.48s
166:	learn: 5.8011917	total: 297ms	remaining: 1.48s
167:	learn: 5.7957299	total: 298ms	remaining: 1.48s
168:	learn: 5.7901204	total: 300ms	remaining: 1.47s
169:	learn: 5.7875583	total: 301ms	remaining: 1.47s
170:	learn: 5.7849508	total: 302ms	remaining: 1.46s
171:	learn: 5.7788432	total: 304ms	remaining: 1.46s
172:	learn: 5.7763170	total: 306ms	remaining: 1.46s
173:	learn: 5.7744379	total: 308ms	remaining: 1.46s
174:	learn: 5.7674005	total: 309ms	remaining: 1.46s
175:	learn: 5.7628535	total: 311ms	remaining: 1.46s
176:	learn: 5.7600787	total: 312ms	remaining: 1.45s
177:	learn: 5.7563065	total: 313ms	remaining: 1.45s
178:	learn: 5.7522971	total: 315ms	remaining: 1.44s
179:	learn: 5.7453555	total: 316ms	remaining: 1.44s
180:	learn: 5.7401068	total: 319ms	remaining: 1.44s
181:	learn: 5.7362873	total: 320ms	remaining: 1.44s
182:	learn: 5.7317286	total: 322ms	remaining: 1.44s
183:	learn: 5.7276720	total: 324ms	remaining: 1.44s
184:	learn: 5.7248626	total: 325ms	remaining: 1.43s
185:	learn: 5.7214478	total: 327ms	remaining: 1.43s
186:	learn: 5.7176332	total: 328ms	remaining: 1.43s
187:	learn: 5.7120950	total: 329ms	remaining: 1.42s
188:	learn: 5.7093204	total: 330ms	remaining: 1.42s
189:	learn: 5.7045701	total: 331ms	remaining: 1.41s
190:	learn: 5.7007636	total: 333ms	remaining: 1.41s
191:	learn: 5.6978587	total: 334ms	remaining: 1.4s
192:	learn: 5.6937738	total: 335ms	remaining: 1.4s
193:	learn: 5.6873147	total: 336ms	remaining: 1.4s
194:	learn: 5.6843210	total: 337ms	remaining: 1.39s
195:	learn: 5.6804800	total: 338ms	remaining: 1.39s
196:	learn: 5.6770665	total: 339ms	remaining: 1.38s
197:	learn: 5.6722370	total: 340ms	remaining: 1.38s
198:	learn: 5.6691557	total: 342ms	remaining: 1.37s
199:	learn: 5.6640227	total: 343ms	remaining: 1.37s
200:	learn: 5.6607241	total: 344ms	remaining: 1.36s
```

```
201:    learn: 5.6570535        total: 345ms    remaining: 1.36s
202:    learn: 5.6543846        total: 346ms    remaining: 1.36s
203:    learn: 5.6494709        total: 347ms    remaining: 1.35s
204:    learn: 5.6433773        total: 348ms    remaining: 1.35s
205:    learn: 5.6374010        total: 348ms    remaining: 1.34s
206:    learn: 5.6341022        total: 349ms    remaining: 1.34s
207:    learn: 5.6303756        total: 350ms    remaining: 1.33s
208:    learn: 5.6284713        total: 351ms    remaining: 1.33s
209:    learn: 5.6254978        total: 352ms    remaining: 1.32s
210:    learn: 5.6224655        total: 353ms    remaining: 1.32s
211:    learn: 5.6173527        total: 354ms    remaining: 1.31s
212:    learn: 5.6136374        total: 355ms    remaining: 1.31s
213:    learn: 5.6108700        total: 356ms    remaining: 1.31s
214:    learn: 5.6070607        total: 357ms    remaining: 1.3s
215:    learn: 5.6042975        total: 359ms    remaining: 1.3s
216:    learn: 5.6000682        total: 360ms    remaining: 1.3s
217:    learn: 5.5965361        total: 361ms    remaining: 1.29s
218:    learn: 5.5910950        total: 362ms    remaining: 1.29s
219:    learn: 5.5864793        total: 363ms    remaining: 1.28s
220:    learn: 5.5828106        total: 364ms    remaining: 1.28s
221:    learn: 5.5812875        total: 364ms    remaining: 1.28s
222:    learn: 5.5786945        total: 365ms    remaining: 1.27s
223:    learn: 5.5741903        total: 366ms    remaining: 1.27s
224:    learn: 5.5714054        total: 367ms    remaining: 1.26s
225:    learn: 5.5672634        total: 368ms    remaining: 1.26s
226:    learn: 5.5646855        total: 369ms    remaining: 1.26s
227:    learn: 5.5618675        total: 370ms    remaining: 1.25s
228:    learn: 5.5583879        total: 371ms    remaining: 1.25s
229:    learn: 5.5558668        total: 372ms    remaining: 1.24s
230:    learn: 5.5529938        total: 373ms    remaining: 1.24s
231:    learn: 5.5500348        total: 374ms    remaining: 1.24s
232:    learn: 5.5465892        total: 375ms    remaining: 1.23s
233:    learn: 5.5423110        total: 376ms    remaining: 1.23s
234:    learn: 5.5397157        total: 377ms    remaining: 1.23s
235:    learn: 5.5351120        total: 377ms    remaining: 1.22s
236:    learn: 5.5303873        total: 378ms    remaining: 1.22s
237:    learn: 5.5264802        total: 379ms    remaining: 1.21s
238:    learn: 5.5233513        total: 380ms    remaining: 1.21s
239:    learn: 5.5208102        total: 381ms    remaining: 1.21s
240:    learn: 5.5182606        total: 382ms    remaining: 1.2s
241:    learn: 5.5136932        total: 383ms    remaining: 1.2s
242:    learn: 5.5087383        total: 384ms    remaining: 1.2s
243:    learn: 5.5064535        total: 385ms    remaining: 1.19s
244:    learn: 5.5036256        total: 386ms    remaining: 1.19s
245:    learn: 5.5017018        total: 387ms    remaining: 1.19s
246:    learn: 5.4969540        total: 388ms    remaining: 1.18s
247:    learn: 5.4934532        total: 389ms    remaining: 1.18s
248:    learn: 5.4888546        total: 390ms    remaining: 1.18s
```

```
249:	learn: 5.4849213	total: 391ms	remaining: 1.17s
250:	learn: 5.4825321	total: 392ms	remaining: 1.17s
251:	learn: 5.4788934	total: 393ms	remaining: 1.17s
252:	learn: 5.4762762	total: 394ms	remaining: 1.16s
253:	learn: 5.4715927	total: 394ms	remaining: 1.16s
254:	learn: 5.4680768	total: 395ms	remaining: 1.16s
255:	learn: 5.4629642	total: 396ms	remaining: 1.15s
256:	learn: 5.4592357	total: 397ms	remaining: 1.15s
257:	learn: 5.4556901	total: 398ms	remaining: 1.15s
258:	learn: 5.4529056	total: 399ms	remaining: 1.14s
259:	learn: 5.4501274	total: 400ms	remaining: 1.14s
260:	learn: 5.4481743	total: 401ms	remaining: 1.14s
261:	learn: 5.4439684	total: 402ms	remaining: 1.13s
262:	learn: 5.4394583	total: 403ms	remaining: 1.13s
263:	learn: 5.4335122	total: 404ms	remaining: 1.13s
264:	learn: 5.4293543	total: 405ms	remaining: 1.12s
265:	learn: 5.4237932	total: 406ms	remaining: 1.12s
266:	learn: 5.4204658	total: 407ms	remaining: 1.12s
267:	learn: 5.4164004	total: 408ms	remaining: 1.11s
268:	learn: 5.4147242	total: 409ms	remaining: 1.11s
269:	learn: 5.4110415	total: 410ms	remaining: 1.11s
270:	learn: 5.4082452	total: 410ms	remaining: 1.1s
271:	learn: 5.4037978	total: 411ms	remaining: 1.1s
272:	learn: 5.4014055	total: 413ms	remaining: 1.1s
273:	learn: 5.3982516	total: 414ms	remaining: 1.09s
274:	learn: 5.3967157	total: 415ms	remaining: 1.09s
275:	learn: 5.3924038	total: 416ms	remaining: 1.09s
276:	learn: 5.3893631	total: 416ms	remaining: 1.09s
277:	learn: 5.3846508	total: 418ms	remaining: 1.08s
278:	learn: 5.3805856	total: 419ms	remaining: 1.08s
279:	learn: 5.3771993	total: 420ms	remaining: 1.08s
280:	learn: 5.3735861	total: 421ms	remaining: 1.08s
281:	learn: 5.3686593	total: 422ms	remaining: 1.07s
282:	learn: 5.3655419	total: 423ms	remaining: 1.07s
283:	learn: 5.3627622	total: 424ms	remaining: 1.07s
284:	learn: 5.3584403	total: 425ms	remaining: 1.07s
285:	learn: 5.3566421	total: 426ms	remaining: 1.06s
286:	learn: 5.3537522	total: 427ms	remaining: 1.06s
287:	learn: 5.3496699	total: 428ms	remaining: 1.06s
288:	learn: 5.3442911	total: 429ms	remaining: 1.05s
289:	learn: 5.3399041	total: 430ms	remaining: 1.05s
290:	learn: 5.3366280	total: 431ms	remaining: 1.05s
291:	learn: 5.3336804	total: 432ms	remaining: 1.05s
292:	learn: 5.3290972	total: 433ms	remaining: 1.04s
293:	learn: 5.3270409	total: 434ms	remaining: 1.04s
294:	learn: 5.3253634	total: 435ms	remaining: 1.04s
295:	learn: 5.3218817	total: 436ms	remaining: 1.04s
296:	learn: 5.3174161	total: 437ms	remaining: 1.03s
```

```
297:    learn: 5.3157057        total: 438ms    remaining: 1.03s
298:    learn: 5.3130667        total: 438ms    remaining: 1.03s
299:    learn: 5.3089667        total: 439ms    remaining: 1.02s
300:    learn: 5.3047717        total: 440ms    remaining: 1.02s
301:    learn: 5.3026334        total: 441ms    remaining: 1.02s
302:    learn: 5.2994596        total: 442ms    remaining: 1.02s
303:    learn: 5.2965309        total: 443ms    remaining: 1.01s
304:    learn: 5.2923769        total: 444ms    remaining: 1.01s
305:    learn: 5.2904734        total: 445ms    remaining: 1.01s
306:    learn: 5.2876843        total: 446ms    remaining: 1.01s
307:    learn: 5.2848087        total: 447ms    remaining: 1s
308:    learn: 5.2816008        total: 448ms    remaining: 1s
309:    learn: 5.2792297        total: 449ms    remaining: 998ms
310:    learn: 5.2741079        total: 450ms    remaining: 996ms
311:    learn: 5.2701139        total: 451ms    remaining: 994ms
312:    learn: 5.2661708        total: 452ms    remaining: 992ms
313:    learn: 5.2647219        total: 453ms    remaining: 989ms
314:    learn: 5.2614116        total: 454ms    remaining: 987ms
315:    learn: 5.2576410        total: 455ms    remaining: 984ms
316:    learn: 5.2555724        total: 456ms    remaining: 982ms
317:    learn: 5.2528320        total: 456ms    remaining: 979ms
318:    learn: 5.2508821        total: 458ms    remaining: 978ms
319:    learn: 5.2469823        total: 459ms    remaining: 976ms
320:    learn: 5.2445004        total: 460ms    remaining: 973ms
321:    learn: 5.2418399        total: 461ms    remaining: 971ms
322:    learn: 5.2387084        total: 462ms    remaining: 969ms
323:    learn: 5.2360976        total: 463ms    remaining: 966ms
324:    learn: 5.2334392        total: 464ms    remaining: 964ms
325:    learn: 5.2305784        total: 465ms    remaining: 962ms
326:    learn: 5.2258672        total: 466ms    remaining: 959ms
327:    learn: 5.2241894        total: 467ms    remaining: 957ms
328:    learn: 5.2210146        total: 468ms    remaining: 954ms
329:    learn: 5.2168267        total: 469ms    remaining: 952ms
330:    learn: 5.2128853        total: 470ms    remaining: 949ms
331:    learn: 5.2100388        total: 471ms    remaining: 947ms
332:    learn: 5.2060544        total: 472ms    remaining: 944ms
333:    learn: 5.2019697        total: 472ms    remaining: 942ms
334:    learn: 5.1988344        total: 473ms    remaining: 939ms
335:    learn: 5.1950748        total: 474ms    remaining: 937ms
336:    learn: 5.1925960        total: 475ms    remaining: 935ms
337:    learn: 5.1906807        total: 477ms    remaining: 934ms
338:    learn: 5.1882085        total: 478ms    remaining: 932ms
339:    learn: 5.1851743        total: 479ms    remaining: 931ms
340:    learn: 5.1806014        total: 481ms    remaining: 929ms
341:    learn: 5.1770670        total: 482ms    remaining: 927ms
342:    learn: 5.1746199        total: 483ms    remaining: 925ms
343:    learn: 5.1702809        total: 484ms    remaining: 923ms
344:    learn: 5.1683796        total: 490ms    remaining: 930ms
```

```
345:    learn: 5.1672715        total: 492ms    remaining: 931ms
346:    learn: 5.1638082        total: 494ms    remaining: 930ms
347:    learn: 5.1593571        total: 496ms    remaining: 930ms
348:    learn: 5.1557889        total: 501ms    remaining: 934ms
349:    learn: 5.1532859        total: 506ms    remaining: 940ms
350:    learn: 5.1506604        total: 512ms    remaining: 947ms
351:    learn: 5.1470752        total: 528ms    remaining: 973ms
352:    learn: 5.1445062        total: 530ms    remaining: 972ms
353:    learn: 5.1412781        total: 533ms    remaining: 972ms
354:    learn: 5.1379650        total: 537ms    remaining: 976ms
355:    learn: 5.1351313        total: 540ms    remaining: 976ms
356:    learn: 5.1314676        total: 542ms    remaining: 976ms
357:    learn: 5.1278864        total: 545ms    remaining: 977ms
358:    learn: 5.1246978        total: 549ms    remaining: 980ms
359:    learn: 5.1224547        total: 551ms    remaining: 980ms
360:    learn: 5.1183759        total: 555ms    remaining: 982ms
361:    learn: 5.1158331        total: 558ms    remaining: 983ms
362:    learn: 5.1129858        total: 560ms    remaining: 983ms
363:    learn: 5.1110267        total: 562ms    remaining: 982ms
364:    learn: 5.1080132        total: 566ms    remaining: 984ms
365:    learn: 5.1047075        total: 570ms    remaining: 988ms
366:    learn: 5.1012734        total: 573ms    remaining: 988ms
367:    learn: 5.0983531        total: 575ms    remaining: 988ms
368:    learn: 5.0959679        total: 578ms    remaining: 989ms
369:    learn: 5.0927453        total: 580ms    remaining: 988ms
370:    learn: 5.0902632        total: 583ms    remaining: 989ms
371:    learn: 5.0876514        total: 586ms    remaining: 989ms
372:    learn: 5.0862853        total: 588ms    remaining: 989ms
373:    learn: 5.0839361        total: 590ms    remaining: 987ms
374:    learn: 5.0799801        total: 591ms    remaining: 986ms
375:    learn: 5.0759489        total: 593ms    remaining: 983ms
376:    learn: 5.0728358        total: 594ms    remaining: 982ms
377:    learn: 5.0706949        total: 596ms    remaining: 980ms
378:    learn: 5.0678286        total: 597ms    remaining: 978ms
379:    learn: 5.0650806        total: 598ms    remaining: 975ms
380:    learn: 5.0623532        total: 599ms    remaining: 973ms
381:    learn: 5.0599611        total: 600ms    remaining: 971ms
382:    learn: 5.0568844        total: 602ms    remaining: 970ms
383:    learn: 5.0538001        total: 603ms    remaining: 968ms
384:    learn: 5.0522454        total: 604ms    remaining: 965ms
385:    learn: 5.0485047        total: 605ms    remaining: 963ms
386:    learn: 5.0448463        total: 607ms    remaining: 962ms
387:    learn: 5.0415277        total: 608ms    remaining: 960ms
388:    learn: 5.0373160        total: 610ms    remaining: 957ms
389:    learn: 5.0343692        total: 611ms    remaining: 955ms
390:    learn: 5.0316565        total: 612ms    remaining: 953ms
391:    learn: 5.0288851        total: 614ms    remaining: 953ms
392:    learn: 5.0255165        total: 616ms    remaining: 951ms
```

```
393:	learn: 5.0228028	total: 617ms	remaining: 949ms
394:	learn: 5.0202408	total: 619ms	remaining: 948ms
395:	learn: 5.0162778	total: 621ms	remaining: 947ms
396:	learn: 5.0133866	total: 622ms	remaining: 945ms
397:	learn: 5.0099075	total: 623ms	remaining: 943ms
398:	learn: 5.0061188	total: 624ms	remaining: 940ms
399:	learn: 5.0027188	total: 625ms	remaining: 938ms
400:	learn: 4.9997221	total: 627ms	remaining: 937ms
401:	learn: 4.9960982	total: 628ms	remaining: 935ms
402:	learn: 4.9935713	total: 630ms	remaining: 933ms
403:	learn: 4.9906951	total: 632ms	remaining: 933ms
404:	learn: 4.9872977	total: 633ms	remaining: 931ms
405:	learn: 4.9822770	total: 635ms	remaining: 929ms
406:	learn: 4.9786810	total: 637ms	remaining: 928ms
407:	learn: 4.9763934	total: 638ms	remaining: 925ms
408:	learn: 4.9739205	total: 639ms	remaining: 923ms
409:	learn: 4.9710993	total: 640ms	remaining: 920ms
410:	learn: 4.9668586	total: 641ms	remaining: 918ms
411:	learn: 4.9636971	total: 642ms	remaining: 916ms
412:	learn: 4.9590642	total: 643ms	remaining: 914ms
413:	learn: 4.9572168	total: 645ms	remaining: 912ms
414:	learn: 4.9547266	total: 646ms	remaining: 911ms
415:	learn: 4.9516503	total: 648ms	remaining: 910ms
416:	learn: 4.9495891	total: 650ms	remaining: 908ms
417:	learn: 4.9461066	total: 651ms	remaining: 906ms
418:	learn: 4.9433669	total: 652ms	remaining: 904ms
419:	learn: 4.9406086	total: 653ms	remaining: 902ms
420:	learn: 4.9376430	total: 655ms	remaining: 901ms
421:	learn: 4.9342531	total: 656ms	remaining: 899ms
422:	learn: 4.9307056	total: 657ms	remaining: 896ms
423:	learn: 4.9280456	total: 658ms	remaining: 894ms
424:	learn: 4.9255807	total: 659ms	remaining: 892ms
425:	learn: 4.9231417	total: 660ms	remaining: 890ms
426:	learn: 4.9195605	total: 662ms	remaining: 888ms
427:	learn: 4.9162949	total: 663ms	remaining: 886ms
428:	learn: 4.9133775	total: 665ms	remaining: 885ms
429:	learn: 4.9110471	total: 669ms	remaining: 886ms
430:	learn: 4.9074843	total: 670ms	remaining: 884ms
431:	learn: 4.9043086	total: 671ms	remaining: 882ms
432:	learn: 4.9018704	total: 672ms	remaining: 880ms
433:	learn: 4.8994685	total: 673ms	remaining: 878ms
434:	learn: 4.8965282	total: 674ms	remaining: 876ms
435:	learn: 4.8938978	total: 675ms	remaining: 874ms
436:	learn: 4.8900264	total: 677ms	remaining: 872ms
437:	learn: 4.8863907	total: 679ms	remaining: 872ms
438:	learn: 4.8840749	total: 681ms	remaining: 870ms
439:	learn: 4.8817455	total: 683ms	remaining: 869ms
440:	learn: 4.8794518	total: 684ms	remaining: 867ms
```

```
441:	learn: 4.8770635	total: 685ms	remaining: 865ms
442:	learn: 4.8738857	total: 686ms	remaining: 863ms
443:	learn: 4.8709771	total: 687ms	remaining: 860ms
444:	learn: 4.8667114	total: 688ms	remaining: 858ms
445:	learn: 4.8636635	total: 690ms	remaining: 856ms
446:	learn: 4.8617771	total: 691ms	remaining: 854ms
447:	learn: 4.8580973	total: 692ms	remaining: 852ms
448:	learn: 4.8555027	total: 693ms	remaining: 850ms
449:	learn: 4.8512357	total: 694ms	remaining: 849ms
450:	learn: 4.8487779	total: 696ms	remaining: 847ms
451:	learn: 4.8461052	total: 698ms	remaining: 846ms
452:	learn: 4.8424439	total: 699ms	remaining: 844ms
453:	learn: 4.8393241	total: 700ms	remaining: 842ms
454:	learn: 4.8374211	total: 702ms	remaining: 840ms
455:	learn: 4.8335157	total: 703ms	remaining: 839ms
456:	learn: 4.8289556	total: 704ms	remaining: 837ms
457:	learn: 4.8271615	total: 705ms	remaining: 835ms
458:	learn: 4.8242517	total: 706ms	remaining: 833ms
459:	learn: 4.8218254	total: 708ms	remaining: 831ms
460:	learn: 4.8200410	total: 709ms	remaining: 829ms
461:	learn: 4.8179711	total: 710ms	remaining: 827ms
462:	learn: 4.8147780	total: 712ms	remaining: 825ms
463:	learn: 4.8111234	total: 713ms	remaining: 824ms
464:	learn: 4.8085756	total: 715ms	remaining: 822ms
465:	learn: 4.8058733	total: 717ms	remaining: 822ms
466:	learn: 4.8041235	total: 718ms	remaining: 820ms
467:	learn: 4.8019816	total: 721ms	remaining: 819ms
468:	learn: 4.7999681	total: 723ms	remaining: 818ms
469:	learn: 4.7975872	total: 724ms	remaining: 817ms
470:	learn: 4.7937582	total: 725ms	remaining: 815ms
471:	learn: 4.7895187	total: 727ms	remaining: 813ms
472:	learn: 4.7872153	total: 729ms	remaining: 812ms
473:	learn: 4.7838314	total: 730ms	remaining: 810ms
474:	learn: 4.7810190	total: 731ms	remaining: 808ms
475:	learn: 4.7796860	total: 732ms	remaining: 806ms
476:	learn: 4.7766431	total: 733ms	remaining: 804ms
477:	learn: 4.7743481	total: 735ms	remaining: 803ms
478:	learn: 4.7726457	total: 736ms	remaining: 801ms
479:	learn: 4.7691990	total: 737ms	remaining: 799ms
480:	learn: 4.7662689	total: 738ms	remaining: 797ms
481:	learn: 4.7631895	total: 740ms	remaining: 795ms
482:	learn: 4.7600672	total: 741ms	remaining: 794ms
483:	learn: 4.7588397	total: 743ms	remaining: 792ms
484:	learn: 4.7555424	total: 744ms	remaining: 790ms
485:	learn: 4.7517052	total: 744ms	remaining: 787ms
486:	learn: 4.7496998	total: 745ms	remaining: 785ms
487:	learn: 4.7474288	total: 746ms	remaining: 783ms
488:	learn: 4.7433962	total: 748ms	remaining: 782ms
```

```
489:	learn: 4.7418905	total: 750ms	remaining: 780ms
490:	learn: 4.7380138	total: 751ms	remaining: 779ms
491:	learn: 4.7355220	total: 752ms	remaining: 777ms
492:	learn: 4.7325920	total: 754ms	remaining: 775ms
493:	learn: 4.7305795	total: 755ms	remaining: 773ms
494:	learn: 4.7290443	total: 756ms	remaining: 771ms
495:	learn: 4.7259906	total: 757ms	remaining: 769ms
496:	learn: 4.7221650	total: 758ms	remaining: 767ms
497:	learn: 4.7189572	total: 759ms	remaining: 765ms
498:	learn: 4.7171112	total: 761ms	remaining: 764ms
499:	learn: 4.7134037	total: 762ms	remaining: 762ms
500:	learn: 4.7105562	total: 763ms	remaining: 760ms
501:	learn: 4.7082851	total: 764ms	remaining: 757ms
502:	learn: 4.7058203	total: 764ms	remaining: 755ms
503:	learn: 4.7023394	total: 766ms	remaining: 753ms
504:	learn: 4.6982019	total: 767ms	remaining: 752ms
505:	learn: 4.6959035	total: 769ms	remaining: 751ms
506:	learn: 4.6920680	total: 770ms	remaining: 749ms
507:	learn: 4.6891146	total: 771ms	remaining: 747ms
508:	learn: 4.6868992	total: 773ms	remaining: 746ms
509:	learn: 4.6851147	total: 774ms	remaining: 744ms
510:	learn: 4.6821829	total: 775ms	remaining: 742ms
511:	learn: 4.6801114	total: 776ms	remaining: 740ms
512:	learn: 4.6765736	total: 777ms	remaining: 738ms
513:	learn: 4.6740866	total: 778ms	remaining: 735ms
514:	learn: 4.6715732	total: 779ms	remaining: 733ms
515:	learn: 4.6684697	total: 780ms	remaining: 732ms
516:	learn: 4.6664006	total: 781ms	remaining: 730ms
517:	learn: 4.6643225	total: 783ms	remaining: 728ms
518:	learn: 4.6630757	total: 784ms	remaining: 726ms
519:	learn: 4.6589788	total: 785ms	remaining: 725ms
520:	learn: 4.6562405	total: 787ms	remaining: 724ms
521:	learn: 4.6543113	total: 788ms	remaining: 722ms
522:	learn: 4.6520457	total: 789ms	remaining: 720ms
523:	learn: 4.6500513	total: 790ms	remaining: 718ms
524:	learn: 4.6465930	total: 791ms	remaining: 716ms
525:	learn: 4.6438202	total: 792ms	remaining: 714ms
526:	learn: 4.6417182	total: 794ms	remaining: 712ms
527:	learn: 4.6389059	total: 795ms	remaining: 711ms
528:	learn: 4.6371039	total: 796ms	remaining: 709ms
529:	learn: 4.6349917	total: 797ms	remaining: 707ms
530:	learn: 4.6321286	total: 799ms	remaining: 706ms
531:	learn: 4.6290795	total: 801ms	remaining: 704ms
532:	learn: 4.6276002	total: 802ms	remaining: 702ms
533:	learn: 4.6243401	total: 803ms	remaining: 701ms
534:	learn: 4.6213682	total: 804ms	remaining: 699ms
535:	learn: 4.6195127	total: 805ms	remaining: 696ms
536:	learn: 4.6182532	total: 806ms	remaining: 695ms
```

```
537:	learn: 4.6156287	total: 808ms	remaining: 693ms
538:	learn: 4.6120018	total: 808ms	remaining: 691ms
539:	learn: 4.6104708	total: 810ms	remaining: 690ms
540:	learn: 4.6075812	total: 811ms	remaining: 688ms
541:	learn: 4.6046223	total: 812ms	remaining: 687ms
542:	learn: 4.6011578	total: 814ms	remaining: 685ms
543:	learn: 4.5984377	total: 815ms	remaining: 683ms
544:	learn: 4.5967512	total: 816ms	remaining: 681ms
545:	learn: 4.5933691	total: 817ms	remaining: 680ms
546:	learn: 4.5908430	total: 819ms	remaining: 678ms
547:	learn: 4.5886341	total: 820ms	remaining: 676ms
548:	learn: 4.5848453	total: 821ms	remaining: 674ms
549:	learn: 4.5829536	total: 822ms	remaining: 672ms
550:	learn: 4.5810620	total: 823ms	remaining: 670ms
551:	learn: 4.5791128	total: 824ms	remaining: 669ms
552:	learn: 4.5772643	total: 825ms	remaining: 667ms
553:	learn: 4.5742337	total: 826ms	remaining: 665ms
554:	learn: 4.5715747	total: 827ms	remaining: 663ms
555:	learn: 4.5696491	total: 829ms	remaining: 662ms
556:	learn: 4.5665298	total: 833ms	remaining: 662ms
557:	learn: 4.5636976	total: 834ms	remaining: 661ms
558:	learn: 4.5614448	total: 835ms	remaining: 659ms
559:	learn: 4.5595029	total: 837ms	remaining: 658ms
560:	learn: 4.5573859	total: 839ms	remaining: 657ms
561:	learn: 4.5534727	total: 840ms	remaining: 655ms
562:	learn: 4.5514557	total: 841ms	remaining: 653ms
563:	learn: 4.5485595	total: 843ms	remaining: 651ms
564:	learn: 4.5453409	total: 845ms	remaining: 650ms
565:	learn: 4.5422307	total: 846ms	remaining: 649ms
566:	learn: 4.5383420	total: 850ms	remaining: 649ms
567:	learn: 4.5360489	total: 852ms	remaining: 648ms
568:	learn: 4.5338724	total: 853ms	remaining: 646ms
569:	learn: 4.5316897	total: 855ms	remaining: 645ms
570:	learn: 4.5298946	total: 857ms	remaining: 644ms
571:	learn: 4.5273324	total: 858ms	remaining: 642ms
572:	learn: 4.5249380	total: 860ms	remaining: 641ms
573:	learn: 4.5225980	total: 862ms	remaining: 640ms
574:	learn: 4.5206559	total: 865ms	remaining: 639ms
575:	learn: 4.5181540	total: 866ms	remaining: 638ms
576:	learn: 4.5158993	total: 868ms	remaining: 637ms
577:	learn: 4.5145179	total: 870ms	remaining: 635ms
578:	learn: 4.5107047	total: 872ms	remaining: 634ms
579:	learn: 4.5092284	total: 873ms	remaining: 632ms
580:	learn: 4.5062966	total: 875ms	remaining: 631ms
581:	learn: 4.5042135	total: 877ms	remaining: 630ms
582:	learn: 4.5011804	total: 879ms	remaining: 629ms
583:	learn: 4.4982436	total: 881ms	remaining: 627ms
584:	learn: 4.4967437	total: 882ms	remaining: 626ms
```

```
585:	learn: 4.4930485	total: 885ms	remaining: 625ms
586:	learn: 4.4915646	total: 887ms	remaining: 624ms
587:	learn: 4.4888834	total: 889ms	remaining: 623ms
588:	learn: 4.4871157	total: 891ms	remaining: 622ms
589:	learn: 4.4854494	total: 895ms	remaining: 622ms
590:	learn: 4.4835968	total: 896ms	remaining: 620ms
591:	learn: 4.4802330	total: 898ms	remaining: 619ms
592:	learn: 4.4782021	total: 899ms	remaining: 617ms
593:	learn: 4.4756610	total: 901ms	remaining: 616ms
594:	learn: 4.4736578	total: 902ms	remaining: 614ms
595:	learn: 4.4707448	total: 904ms	remaining: 613ms
596:	learn: 4.4683522	total: 907ms	remaining: 612ms
597:	learn: 4.4658591	total: 908ms	remaining: 611ms
598:	learn: 4.4635074	total: 909ms	remaining: 609ms
599:	learn: 4.4613261	total: 911ms	remaining: 608ms
600:	learn: 4.4575209	total: 913ms	remaining: 606ms
601:	learn: 4.4559858	total: 914ms	remaining: 604ms
602:	learn: 4.4537301	total: 915ms	remaining: 603ms
603:	learn: 4.4519200	total: 916ms	remaining: 601ms
604:	learn: 4.4488924	total: 918ms	remaining: 599ms
605:	learn: 4.4454451	total: 919ms	remaining: 598ms
606:	learn: 4.4436782	total: 920ms	remaining: 596ms
607:	learn: 4.4406245	total: 922ms	remaining: 594ms
608:	learn: 4.4370080	total: 923ms	remaining: 592ms
609:	learn: 4.4352391	total: 924ms	remaining: 591ms
610:	learn: 4.4327956	total: 925ms	remaining: 589ms
611:	learn: 4.4304466	total: 926ms	remaining: 587ms
612:	learn: 4.4277766	total: 927ms	remaining: 585ms
613:	learn: 4.4242440	total: 928ms	remaining: 584ms
614:	learn: 4.4223065	total: 929ms	remaining: 582ms
615:	learn: 4.4200214	total: 931ms	remaining: 581ms
616:	learn: 4.4183431	total: 932ms	remaining: 579ms
617:	learn: 4.4156633	total: 934ms	remaining: 577ms
618:	learn: 4.4138517	total: 935ms	remaining: 575ms
619:	learn: 4.4117497	total: 936ms	remaining: 573ms
620:	learn: 4.4076910	total: 937ms	remaining: 572ms
621:	learn: 4.4057668	total: 938ms	remaining: 570ms
622:	learn: 4.4042259	total: 939ms	remaining: 568ms
623:	learn: 4.4025635	total: 940ms	remaining: 566ms
624:	learn: 4.4008409	total: 941ms	remaining: 565ms
625:	learn: 4.3992273	total: 942ms	remaining: 563ms
626:	learn: 4.3955086	total: 943ms	remaining: 561ms
627:	learn: 4.3933511	total: 944ms	remaining: 559ms
628:	learn: 4.3907497	total: 945ms	remaining: 558ms
629:	learn: 4.3889763	total: 946ms	remaining: 556ms
630:	learn: 4.3851939	total: 947ms	remaining: 554ms
631:	learn: 4.3825361	total: 948ms	remaining: 552ms
632:	learn: 4.3812838	total: 949ms	remaining: 550ms
```

```
633:	learn: 4.3793939	total: 950ms	remaining: 548ms
634:	learn: 4.3770202	total: 951ms	remaining: 547ms
635:	learn: 4.3753107	total: 952ms	remaining: 545ms
636:	learn: 4.3720902	total: 953ms	remaining: 543ms
637:	learn: 4.3690731	total: 954ms	remaining: 541ms
638:	learn: 4.3661967	total: 955ms	remaining: 539ms
639:	learn: 4.3635800	total: 956ms	remaining: 538ms
640:	learn: 4.3603460	total: 957ms	remaining: 536ms
641:	learn: 4.3570057	total: 958ms	remaining: 534ms
642:	learn: 4.3542237	total: 959ms	remaining: 532ms
643:	learn: 4.3524326	total: 960ms	remaining: 531ms
644:	learn: 4.3508514	total: 961ms	remaining: 529ms
645:	learn: 4.3486219	total: 962ms	remaining: 527ms
646:	learn: 4.3456551	total: 963ms	remaining: 525ms
647:	learn: 4.3423854	total: 964ms	remaining: 524ms
648:	learn: 4.3401817	total: 965ms	remaining: 522ms
649:	learn: 4.3387503	total: 966ms	remaining: 520ms
650:	learn: 4.3366942	total: 967ms	remaining: 519ms
651:	learn: 4.3339186	total: 968ms	remaining: 517ms
652:	learn: 4.3328043	total: 969ms	remaining: 515ms
653:	learn: 4.3294056	total: 970ms	remaining: 513ms
654:	learn: 4.3275942	total: 971ms	remaining: 512ms
655:	learn: 4.3253965	total: 972ms	remaining: 510ms
656:	learn: 4.3239030	total: 973ms	remaining: 508ms
657:	learn: 4.3224368	total: 974ms	remaining: 506ms
658:	learn: 4.3205104	total: 975ms	remaining: 505ms
659:	learn: 4.3190299	total: 977ms	remaining: 503ms
660:	learn: 4.3171059	total: 978ms	remaining: 501ms
661:	learn: 4.3134324	total: 979ms	remaining: 500ms
662:	learn: 4.3114060	total: 979ms	remaining: 498ms
663:	learn: 4.3090055	total: 980ms	remaining: 496ms
664:	learn: 4.3060282	total: 982ms	remaining: 495ms
665:	learn: 4.3040665	total: 983ms	remaining: 493ms
666:	learn: 4.3010973	total: 984ms	remaining: 491ms
667:	learn: 4.2988361	total: 985ms	remaining: 490ms
668:	learn: 4.2967871	total: 986ms	remaining: 488ms
669:	learn: 4.2938858	total: 987ms	remaining: 486ms
670:	learn: 4.2928340	total: 988ms	remaining: 484ms
671:	learn: 4.2914117	total: 989ms	remaining: 483ms
672:	learn: 4.2884116	total: 990ms	remaining: 481ms
673:	learn: 4.2850292	total: 992ms	remaining: 480ms
674:	learn: 4.2826040	total: 993ms	remaining: 478ms
675:	learn: 4.2808913	total: 994ms	remaining: 476ms
676:	learn: 4.2788215	total: 995ms	remaining: 475ms
677:	learn: 4.2778401	total: 996ms	remaining: 473ms
678:	learn: 4.2752736	total: 997ms	remaining: 471ms
679:	learn: 4.2731982	total: 998ms	remaining: 470ms
680:	learn: 4.2710262	total: 999ms	remaining: 468ms
```

```
681:    learn: 4.2683997        total: 1000ms   remaining: 466ms
682:    learn: 4.2651731        total: 1s       remaining: 465ms
683:    learn: 4.2631976        total: 1s       remaining: 463ms
684:    learn: 4.2613667        total: 1s       remaining: 461ms
685:    learn: 4.2595549        total: 1s       remaining: 459ms
686:    learn: 4.2568476        total: 1s       remaining: 458ms
687:    learn: 4.2555659        total: 1s       remaining: 456ms
688:    learn: 4.2525124        total: 1.01s    remaining: 454ms
689:    learn: 4.2499586        total: 1.01s    remaining: 453ms
690:    learn: 4.2485145        total: 1.01s    remaining: 451ms
691:    learn: 4.2465194        total: 1.01s    remaining: 450ms
692:    learn: 4.2446787        total: 1.01s    remaining: 448ms
693:    learn: 4.2431712        total: 1.01s    remaining: 446ms
694:    learn: 4.2415673        total: 1.01s    remaining: 444ms
695:    learn: 4.2398202        total: 1.01s    remaining: 443ms
696:    learn: 4.2364895        total: 1.01s    remaining: 441ms
697:    learn: 4.2338837        total: 1.01s    remaining: 439ms
698:    learn: 4.2324389        total: 1.02s    remaining: 438ms
699:    learn: 4.2295222        total: 1.02s    remaining: 436ms
700:    learn: 4.2277050        total: 1.02s    remaining: 435ms
701:    learn: 4.2259565        total: 1.02s    remaining: 433ms
702:    learn: 4.2230299        total: 1.02s    remaining: 431ms
703:    learn: 4.2203086        total: 1.02s    remaining: 430ms
704:    learn: 4.2179995        total: 1.02s    remaining: 428ms
705:    learn: 4.2154331        total: 1.02s    remaining: 426ms
706:    learn: 4.2131152        total: 1.02s    remaining: 425ms
707:    learn: 4.2097552        total: 1.02s    remaining: 423ms
708:    learn: 4.2088140        total: 1.03s    remaining: 421ms
709:    learn: 4.2070385        total: 1.03s    remaining: 420ms
710:    learn: 4.2039257        total: 1.03s    remaining: 419ms
711:    learn: 4.2009112        total: 1.03s    remaining: 417ms
712:    learn: 4.1988530        total: 1.03s    remaining: 415ms
713:    learn: 4.1954322        total: 1.03s    remaining: 413ms
714:    learn: 4.1923651        total: 1.03s    remaining: 412ms
715:    learn: 4.1910812        total: 1.03s    remaining: 411ms
716:    learn: 4.1879048        total: 1.04s    remaining: 409ms
717:    learn: 4.1864327        total: 1.04s    remaining: 408ms
718:    learn: 4.1847341        total: 1.04s    remaining: 406ms
719:    learn: 4.1825321        total: 1.04s    remaining: 405ms
720:    learn: 4.1807066        total: 1.04s    remaining: 403ms
721:    learn: 4.1790232        total: 1.04s    remaining: 402ms
722:    learn: 4.1774890        total: 1.04s    remaining: 400ms
723:    learn: 4.1755582        total: 1.04s    remaining: 398ms
724:    learn: 4.1729859        total: 1.04s    remaining: 397ms
725:    learn: 4.1710243        total: 1.05s    remaining: 395ms
726:    learn: 4.1695078        total: 1.05s    remaining: 394ms
727:    learn: 4.1681219        total: 1.05s    remaining: 392ms
728:    learn: 4.1667232        total: 1.05s    remaining: 391ms
```

```
729:	learn: 4.1648470	total: 1.05s	remaining: 389ms
730:	learn: 4.1628405	total: 1.05s	remaining: 387ms
731:	learn: 4.1612227	total: 1.05s	remaining: 386ms
732:	learn: 4.1597834	total: 1.05s	remaining: 385ms
733:	learn: 4.1576790	total: 1.06s	remaining: 383ms
734:	learn: 4.1544970	total: 1.06s	remaining: 381ms
735:	learn: 4.1517667	total: 1.06s	remaining: 380ms
736:	learn: 4.1502781	total: 1.06s	remaining: 378ms
737:	learn: 4.1473730	total: 1.06s	remaining: 377ms
738:	learn: 4.1435993	total: 1.06s	remaining: 376ms
739:	learn: 4.1418034	total: 1.06s	remaining: 374ms
740:	learn: 4.1398163	total: 1.06s	remaining: 372ms
741:	learn: 4.1383439	total: 1.07s	remaining: 371ms
742:	learn: 4.1359939	total: 1.07s	remaining: 370ms
743:	learn: 4.1339952	total: 1.07s	remaining: 368ms
744:	learn: 4.1324095	total: 1.07s	remaining: 366ms
745:	learn: 4.1312680	total: 1.07s	remaining: 365ms
746:	learn: 4.1301179	total: 1.07s	remaining: 363ms
747:	learn: 4.1273495	total: 1.07s	remaining: 361ms
748:	learn: 4.1253710	total: 1.07s	remaining: 360ms
749:	learn: 4.1231733	total: 1.07s	remaining: 358ms
750:	learn: 4.1215212	total: 1.08s	remaining: 357ms
751:	learn: 4.1187178	total: 1.08s	remaining: 355ms
752:	learn: 4.1165180	total: 1.08s	remaining: 354ms
753:	learn: 4.1151850	total: 1.08s	remaining: 352ms
754:	learn: 4.1135856	total: 1.08s	remaining: 350ms
755:	learn: 4.1122660	total: 1.08s	remaining: 349ms
756:	learn: 4.1099398	total: 1.08s	remaining: 347ms
757:	learn: 4.1083219	total: 1.08s	remaining: 346ms
758:	learn: 4.1072764	total: 1.08s	remaining: 344ms
759:	learn: 4.1061600	total: 1.08s	remaining: 343ms
760:	learn: 4.1036455	total: 1.09s	remaining: 341ms
761:	learn: 4.1003609	total: 1.09s	remaining: 340ms
762:	learn: 4.0977573	total: 1.09s	remaining: 338ms
763:	learn: 4.0958283	total: 1.09s	remaining: 337ms
764:	learn: 4.0935881	total: 1.09s	remaining: 335ms
765:	learn: 4.0919933	total: 1.09s	remaining: 333ms
766:	learn: 4.0900928	total: 1.09s	remaining: 332ms
767:	learn: 4.0878078	total: 1.09s	remaining: 331ms
768:	learn: 4.0853244	total: 1.09s	remaining: 329ms
769:	learn: 4.0825169	total: 1.09s	remaining: 327ms
770:	learn: 4.0799294	total: 1.1s	remaining: 326ms
771:	learn: 4.0771666	total: 1.1s	remaining: 324ms
772:	learn: 4.0749992	total: 1.1s	remaining: 323ms
773:	learn: 4.0728754	total: 1.1s	remaining: 321ms
774:	learn: 4.0711646	total: 1.1s	remaining: 320ms
775:	learn: 4.0697692	total: 1.1s	remaining: 318ms
776:	learn: 4.0677538	total: 1.1s	remaining: 317ms
```

```
777:	learn: 4.0665245	total: 1.1s	remaining: 315ms
778:	learn: 4.0650785	total: 1.11s	remaining: 314ms
779:	learn: 4.0635928	total: 1.11s	remaining: 312ms
780:	learn: 4.0622841	total: 1.11s	remaining: 311ms
781:	learn: 4.0605165	total: 1.11s	remaining: 309ms
782:	learn: 4.0591472	total: 1.11s	remaining: 308ms
783:	learn: 4.0567405	total: 1.11s	remaining: 306ms
784:	learn: 4.0558828	total: 1.11s	remaining: 305ms
785:	learn: 4.0541034	total: 1.11s	remaining: 303ms
786:	learn: 4.0513490	total: 1.11s	remaining: 302ms
787:	learn: 4.0501228	total: 1.11s	remaining: 300ms
788:	learn: 4.0477523	total: 1.12s	remaining: 299ms
789:	learn: 4.0460243	total: 1.12s	remaining: 297ms
790:	learn: 4.0447685	total: 1.12s	remaining: 295ms
791:	learn: 4.0421851	total: 1.12s	remaining: 294ms
792:	learn: 4.0410712	total: 1.12s	remaining: 293ms
793:	learn: 4.0380422	total: 1.12s	remaining: 291ms
794:	learn: 4.0355865	total: 1.12s	remaining: 289ms
795:	learn: 4.0322316	total: 1.12s	remaining: 288ms
796:	learn: 4.0299251	total: 1.12s	remaining: 286ms
797:	learn: 4.0280671	total: 1.13s	remaining: 285ms
798:	learn: 4.0263122	total: 1.13s	remaining: 283ms
799:	learn: 4.0250807	total: 1.13s	remaining: 282ms
800:	learn: 4.0223998	total: 1.13s	remaining: 280ms
801:	learn: 4.0214262	total: 1.13s	remaining: 279ms
802:	learn: 4.0192259	total: 1.13s	remaining: 277ms
803:	learn: 4.0171559	total: 1.13s	remaining: 276ms
804:	learn: 4.0151516	total: 1.13s	remaining: 275ms
805:	learn: 4.0132433	total: 1.13s	remaining: 273ms
806:	learn: 4.0110134	total: 1.14s	remaining: 272ms
807:	learn: 4.0094213	total: 1.14s	remaining: 270ms
808:	learn: 4.0085255	total: 1.14s	remaining: 269ms
809:	learn: 4.0074637	total: 1.14s	remaining: 267ms
810:	learn: 4.0052995	total: 1.14s	remaining: 266ms
811:	learn: 4.0025456	total: 1.14s	remaining: 264ms
812:	learn: 4.0015187	total: 1.14s	remaining: 263ms
813:	learn: 4.0001356	total: 1.14s	remaining: 261ms
814:	learn: 3.9995160	total: 1.14s	remaining: 260ms
815:	learn: 3.9977811	total: 1.15s	remaining: 258ms
816:	learn: 3.9948776	total: 1.15s	remaining: 257ms
817:	learn: 3.9926629	total: 1.15s	remaining: 255ms
818:	learn: 3.9912591	total: 1.15s	remaining: 254ms
819:	learn: 3.9893617	total: 1.15s	remaining: 252ms
820:	learn: 3.9869705	total: 1.15s	remaining: 251ms
821:	learn: 3.9821527	total: 1.15s	remaining: 250ms
822:	learn: 3.9801054	total: 1.15s	remaining: 248ms
823:	learn: 3.9787040	total: 1.15s	remaining: 247ms
824:	learn: 3.9772015	total: 1.16s	remaining: 245ms
```

```
825:	learn: 3.9758180	total: 1.16s	remaining: 244ms
826:	learn: 3.9746737	total: 1.16s	remaining: 242ms
827:	learn: 3.9731466	total: 1.16s	remaining: 241ms
828:	learn: 3.9716880	total: 1.16s	remaining: 239ms
829:	learn: 3.9696691	total: 1.16s	remaining: 238ms
830:	learn: 3.9686138	total: 1.16s	remaining: 236ms
831:	learn: 3.9660036	total: 1.16s	remaining: 235ms
832:	learn: 3.9623518	total: 1.16s	remaining: 233ms
833:	learn: 3.9612115	total: 1.17s	remaining: 232ms
834:	learn: 3.9599351	total: 1.17s	remaining: 230ms
835:	learn: 3.9580834	total: 1.17s	remaining: 229ms
836:	learn: 3.9566679	total: 1.17s	remaining: 228ms
837:	learn: 3.9546921	total: 1.17s	remaining: 226ms
838:	learn: 3.9522099	total: 1.17s	remaining: 225ms
839:	learn: 3.9507194	total: 1.17s	remaining: 223ms
840:	learn: 3.9484403	total: 1.17s	remaining: 222ms
841:	learn: 3.9466429	total: 1.17s	remaining: 220ms
842:	learn: 3.9449331	total: 1.17s	remaining: 219ms
843:	learn: 3.9440681	total: 1.17s	remaining: 217ms
844:	learn: 3.9419885	total: 1.18s	remaining: 216ms
845:	learn: 3.9406260	total: 1.18s	remaining: 214ms
846:	learn: 3.9387902	total: 1.18s	remaining: 213ms
847:	learn: 3.9376630	total: 1.18s	remaining: 211ms
848:	learn: 3.9346902	total: 1.18s	remaining: 210ms
849:	learn: 3.9328289	total: 1.18s	remaining: 208ms
850:	learn: 3.9305284	total: 1.18s	remaining: 207ms
851:	learn: 3.9285110	total: 1.18s	remaining: 205ms
852:	learn: 3.9267236	total: 1.18s	remaining: 204ms
853:	learn: 3.9250028	total: 1.18s	remaining: 202ms
854:	learn: 3.9205846	total: 1.19s	remaining: 201ms
855:	learn: 3.9188683	total: 1.19s	remaining: 200ms
856:	learn: 3.9177162	total: 1.19s	remaining: 198ms
857:	learn: 3.9148179	total: 1.19s	remaining: 197ms
858:	learn: 3.9138677	total: 1.19s	remaining: 195ms
859:	learn: 3.9112682	total: 1.19s	remaining: 194ms
860:	learn: 3.9095277	total: 1.19s	remaining: 192ms
861:	learn: 3.9083598	total: 1.19s	remaining: 191ms
862:	learn: 3.9065219	total: 1.19s	remaining: 189ms
863:	learn: 3.9042369	total: 1.19s	remaining: 188ms
864:	learn: 3.9021938	total: 1.19s	remaining: 186ms
865:	learn: 3.9010610	total: 1.2s	remaining: 185ms
866:	learn: 3.8996989	total: 1.2s	remaining: 184ms
867:	learn: 3.8983895	total: 1.2s	remaining: 182ms
868:	learn: 3.8970045	total: 1.2s	remaining: 181ms
869:	learn: 3.8957021	total: 1.2s	remaining: 179ms
870:	learn: 3.8939657	total: 1.2s	remaining: 178ms
871:	learn: 3.8905694	total: 1.2s	remaining: 177ms
872:	learn: 3.8883995	total: 1.2s	remaining: 175ms
```

```
873:	learn: 3.8860372	total: 1.2s	remaining: 174ms
874:	learn: 3.8845084	total: 1.21s	remaining: 172ms
875:	learn: 3.8832428	total: 1.21s	remaining: 171ms
876:	learn: 3.8804895	total: 1.21s	remaining: 169ms
877:	learn: 3.8791703	total: 1.21s	remaining: 168ms
878:	learn: 3.8777256	total: 1.21s	remaining: 167ms
879:	learn: 3.8761806	total: 1.21s	remaining: 165ms
880:	learn: 3.8729513	total: 1.21s	remaining: 164ms
881:	learn: 3.8711632	total: 1.21s	remaining: 162ms
882:	learn: 3.8693628	total: 1.21s	remaining: 161ms
883:	learn: 3.8652249	total: 1.22s	remaining: 160ms
884:	learn: 3.8641669	total: 1.22s	remaining: 158ms
885:	learn: 3.8624438	total: 1.22s	remaining: 157ms
886:	learn: 3.8604975	total: 1.22s	remaining: 155ms
887:	learn: 3.8594698	total: 1.22s	remaining: 154ms
888:	learn: 3.8571502	total: 1.22s	remaining: 153ms
889:	learn: 3.8548837	total: 1.22s	remaining: 151ms
890:	learn: 3.8523085	total: 1.22s	remaining: 150ms
891:	learn: 3.8507946	total: 1.23s	remaining: 148ms
892:	learn: 3.8491568	total: 1.23s	remaining: 147ms
893:	learn: 3.8467799	total: 1.23s	remaining: 146ms
894:	learn: 3.8454886	total: 1.23s	remaining: 144ms
895:	learn: 3.8437002	total: 1.23s	remaining: 143ms
896:	learn: 3.8415643	total: 1.23s	remaining: 141ms
897:	learn: 3.8394229	total: 1.23s	remaining: 140ms
898:	learn: 3.8379936	total: 1.23s	remaining: 138ms
899:	learn: 3.8367794	total: 1.23s	remaining: 137ms
900:	learn: 3.8344801	total: 1.23s	remaining: 136ms
901:	learn: 3.8320083	total: 1.24s	remaining: 134ms
902:	learn: 3.8306925	total: 1.24s	remaining: 133ms
903:	learn: 3.8284992	total: 1.24s	remaining: 132ms
904:	learn: 3.8270479	total: 1.24s	remaining: 130ms
905:	learn: 3.8249230	total: 1.24s	remaining: 129ms
906:	learn: 3.8236542	total: 1.24s	remaining: 127ms
907:	learn: 3.8211605	total: 1.24s	remaining: 126ms
908:	learn: 3.8203677	total: 1.24s	remaining: 124ms
909:	learn: 3.8181358	total: 1.24s	remaining: 123ms
910:	learn: 3.8166697	total: 1.25s	remaining: 122ms
911:	learn: 3.8142736	total: 1.25s	remaining: 120ms
912:	learn: 3.8130584	total: 1.25s	remaining: 119ms
913:	learn: 3.8103382	total: 1.25s	remaining: 118ms
914:	learn: 3.8092852	total: 1.25s	remaining: 116ms
915:	learn: 3.8074425	total: 1.25s	remaining: 115ms
916:	learn: 3.8047195	total: 1.25s	remaining: 113ms
917:	learn: 3.8017577	total: 1.25s	remaining: 112ms
918:	learn: 3.7997481	total: 1.26s	remaining: 111ms
919:	learn: 3.7988910	total: 1.26s	remaining: 109ms
920:	learn: 3.7980989	total: 1.26s	remaining: 108ms
```

```
921:	learn: 3.7960518	total: 1.26s	remaining: 107ms
922:	learn: 3.7945343	total: 1.26s	remaining: 105ms
923:	learn: 3.7935557	total: 1.26s	remaining: 104ms
924:	learn: 3.7925738	total: 1.26s	remaining: 102ms
925:	learn: 3.7896580	total: 1.26s	remaining: 101ms
926:	learn: 3.7877780	total: 1.26s	remaining: 99.6ms
927:	learn: 3.7863220	total: 1.26s	remaining: 98.2ms
928:	learn: 3.7839397	total: 1.27s	remaining: 96.8ms
929:	learn: 3.7822853	total: 1.27s	remaining: 95.4ms
930:	learn: 3.7807782	total: 1.27s	remaining: 94ms
931:	learn: 3.7786154	total: 1.27s	remaining: 92.6ms
932:	learn: 3.7767656	total: 1.27s	remaining: 91.3ms
933:	learn: 3.7743122	total: 1.27s	remaining: 89.9ms
934:	learn: 3.7721269	total: 1.27s	remaining: 88.5ms
935:	learn: 3.7711732	total: 1.27s	remaining: 87.1ms
936:	learn: 3.7681714	total: 1.27s	remaining: 85.7ms
937:	learn: 3.7660847	total: 1.27s	remaining: 84.3ms
938:	learn: 3.7647763	total: 1.28s	remaining: 82.9ms
939:	learn: 3.7625258	total: 1.28s	remaining: 81.5ms
940:	learn: 3.7606396	total: 1.28s	remaining: 80.2ms
941:	learn: 3.7585125	total: 1.28s	remaining: 78.8ms
942:	learn: 3.7564207	total: 1.28s	remaining: 77.4ms
943:	learn: 3.7546091	total: 1.28s	remaining: 76ms
944:	learn: 3.7532327	total: 1.28s	remaining: 74.6ms
945:	learn: 3.7521011	total: 1.28s	remaining: 73.2ms
946:	learn: 3.7506047	total: 1.28s	remaining: 71.8ms
947:	learn: 3.7483719	total: 1.28s	remaining: 70.5ms
948:	learn: 3.7459028	total: 1.28s	remaining: 69.1ms
949:	learn: 3.7441130	total: 1.29s	remaining: 67.7ms
950:	learn: 3.7424897	total: 1.29s	remaining: 66.3ms
951:	learn: 3.7405450	total: 1.29s	remaining: 64.9ms
952:	learn: 3.7397920	total: 1.29s	remaining: 63.6ms
953:	learn: 3.7387401	total: 1.29s	remaining: 62.2ms
954:	learn: 3.7376545	total: 1.29s	remaining: 60.8ms
955:	learn: 3.7363115	total: 1.29s	remaining: 59.5ms
956:	learn: 3.7340187	total: 1.29s	remaining: 58.1ms
957:	learn: 3.7325666	total: 1.29s	remaining: 56.7ms
958:	learn: 3.7310458	total: 1.29s	remaining: 55.4ms
959:	learn: 3.7290430	total: 1.29s	remaining: 54ms
960:	learn: 3.7278364	total: 1.3s	remaining: 52.6ms
961:	learn: 3.7263543	total: 1.3s	remaining: 51.3ms
962:	learn: 3.7249360	total: 1.3s	remaining: 49.9ms
963:	learn: 3.7232797	total: 1.3s	remaining: 48.5ms
964:	learn: 3.7207720	total: 1.3s	remaining: 47.2ms
965:	learn: 3.7190951	total: 1.3s	remaining: 45.8ms
966:	learn: 3.7182209	total: 1.3s	remaining: 44.4ms
967:	learn: 3.7167485	total: 1.3s	remaining: 43.1ms
968:	learn: 3.7156957	total: 1.3s	remaining: 41.7ms
```

```
969:	learn: 3.7144164	total: 1.3s	remaining: 40.4ms
970:	learn: 3.7122265	total: 1.3s	remaining: 39ms
971:	learn: 3.7108344	total: 1.31s	remaining: 37.6ms
972:	learn: 3.7095659	total: 1.31s	remaining: 36.3ms
973:	learn: 3.7084918	total: 1.31s	remaining: 34.9ms
974:	learn: 3.7072838	total: 1.31s	remaining: 33.6ms
975:	learn: 3.7054338	total: 1.31s	remaining: 32.2ms
976:	learn: 3.7042914	total: 1.31s	remaining: 30.9ms
977:	learn: 3.7025495	total: 1.31s	remaining: 29.5ms
978:	learn: 3.7011849	total: 1.31s	remaining: 28.2ms
979:	learn: 3.6992305	total: 1.31s	remaining: 26.8ms
980:	learn: 3.6975253	total: 1.31s	remaining: 25.5ms
981:	learn: 3.6963953	total: 1.31s	remaining: 24.1ms
982:	learn: 3.6949483	total: 1.32s	remaining: 22.8ms
983:	learn: 3.6920328	total: 1.32s	remaining: 21.4ms
984:	learn: 3.6893231	total: 1.32s	remaining: 20.1ms
985:	learn: 3.6867281	total: 1.32s	remaining: 18.7ms
986:	learn: 3.6849608	total: 1.32s	remaining: 17.4ms
987:	learn: 3.6823086	total: 1.32s	remaining: 16.1ms
988:	learn: 3.6805610	total: 1.32s	remaining: 14.7ms
989:	learn: 3.6779277	total: 1.32s	remaining: 13.4ms
990:	learn: 3.6765135	total: 1.32s	remaining: 12ms
991:	learn: 3.6748904	total: 1.32s	remaining: 10.7ms
992:	learn: 3.6737811	total: 1.33s	remaining: 9.35ms
993:	learn: 3.6719689	total: 1.33s	remaining: 8.01ms
994:	learn: 3.6703036	total: 1.33s	remaining: 6.67ms
995:	learn: 3.6682224	total: 1.33s	remaining: 5.34ms
996:	learn: 3.6666566	total: 1.33s	remaining: 4ms
997:	learn: 3.6648276	total: 1.33s	remaining: 2.67ms
998:	learn: 3.6639379	total: 1.33s	remaining: 1.33ms
999:	learn: 3.6628745	total: 1.33s	remaining: 0us
XGBoost Mean Squared Error (MSE): 55.5051088664682
XGBoost Root Mean Squared Error (RMSE): 7.4501750896517995
LightBoost Mean Squared Error (MSE): 48.157920350243195
LightBoost Root Mean Squared Error (RMSE): 6.939590791267393
CatBoost Mean Squared Error (MSE): 49.48130042521675
CatBoost Root Mean Squared Error (RMSE): 7.034294593291978
```

[142]:

```
#  make a lasso model
from sklearn.linear_model import Lasso

# Initialize the model
lasso_model = Lasso()

# Train the model
lasso_model.fit(X_train, y_train)
```

```python
# Predict using the model
lasso_predictions = lasso_model.predict(X_test)

# Evaluating model performance
lasso_mse = mean_squared_error(y_test, lasso_predictions)
lasso_rmse = np.sqrt(lasso_mse)

print(f"Lasso Mean Squared Error (MSE): {lasso_mse}")
print(f"Lasso Root Mean Squared Error (RMSE): {lasso_rmse}")
```

```
Lasso Mean Squared Error (MSE): 55.32508360399317
Lasso Root Mean Squared Error (RMSE): 7.438083328653502
```

[143]:
```python
# make a KNN model
from sklearn.neighbors import KNeighborsRegressor


# Initialize the model
knn_model = KNeighborsRegressor()

# Train the model
knn_model.fit(X_train, y_train)

# Predict using the model
knn_predictions = knn_model.predict(X_test)

# Evaluating model performance
knn_mse = mean_squared_error(y_test, knn_predictions)
knn_rmse = np.sqrt(knn_mse)

print(f"KNN Mean Squared Error (MSE): {knn_mse}")
print(f"KNN Root Mean Squared Error (RMSE): {knn_rmse}")
```

```
KNN Mean Squared Error (MSE): 53.123042755878316
KNN Root Mean Squared Error (RMSE): 7.28855560148088
```

[144]:
```python
# make a gaussian process model
from sklearn.gaussian_process import GaussianProcessRegressor

# Initialize the model
gp_model = GaussianProcessRegressor()

# Train the model
gp_model.fit(X_train, y_train)

# Predict using the model
gp_predictions = gp_model.predict(X_test)
```

```
# Evaluating model performance
gp_mse = mean_squared_error(y_test, gp_predictions)
gp_rmse = np.sqrt(gp_mse)

print(f"Gaussian Process Mean Squared Error (MSE): {gp_mse}")
print(f"Gaussian Process Root Mean Squared Error (RMSE): {gp_rmse}")
```

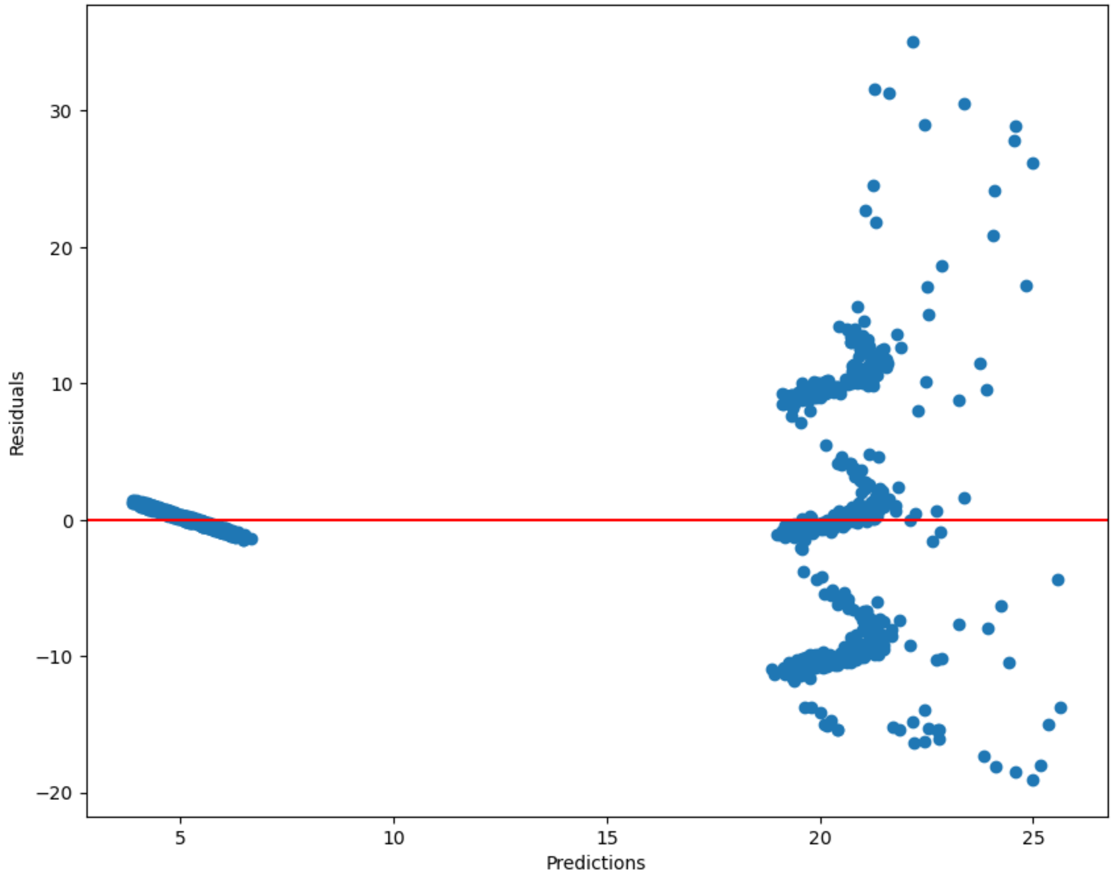
```
Gaussian Process Mean Squared Error (MSE): 134.2358197540432
Gaussian Process Root Mean Squared Error (RMSE): 11.586018287316968
```

[162]:
```
# make a bayesian ridge model
from sklearn.linear_model import BayesianRidge
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Initialize the model
bayes_model = BayesianRidge()

# Train the model
bayes_model.fit(X_train, y_train)

# Predict using the model
bayes_predictions = bayes_model.predict(X_test)

# Evaluating model performance
bayes_mse = mean_squared_error(y_test, bayes_predictions)
bayes_rmse = np.sqrt(bayes_mse)
bayes_mae = mean_absolute_error(y_test, bayes_predictions)
bayes_r_squared = r2_score(y_test, bayes_predictions)

print(f"Bayesian Ridge Mean Squared Error (MSE): {bayes_mse}")
print(f"Bayesian Ridge Root Mean Squared Error (RMSE): {bayes_rmse}")
print(f"Bayesian Ridge Mean Absolute Error (MAE): {bayes_mae}")
print(f"Bayesian Ridge R2 Score: {bayes_r_squared}")
```

```
Bayesian Ridge Mean Squared Error (MSE): 47.334828584133646
Bayesian Ridge Root Mean Squared Error (RMSE): 6.8800311470322315
Bayesian Ridge Mean Absolute Error (MAE): 4.212244922958837
Bayesian Ridge R2 Score: 0.5687861927524951
```

[163]:
```
# my best model is the bayes regression ans so I will use it to make␣
↪predictions on the test_2.csv file. Before that, plot residuals
# Plotting residuals
plt.figure(figsize=(10, 8))
plt.scatter(x=bayes_predictions, y=y_test - bayes_predictions)
plt.xlabel('Predictions')
plt.ylabel('Residuals')
```

```
plt.axhline(y=0, color='r', linestyle='-')
plt.show()
```



[164]:
```
# with my test_df as my X, predict the shipping time
bayes_test_predictions = bayes_model.predict(test_df)

# print the predictions
print(bayes_test_predictions)
```

```
[ 5.30096977  4.86433048  4.79073452 … 22.67933239 23.81081131
 21.59474882]
```

[165]:
```
bayes_test_predictions = pd.DataFrame(pd.read_csv('test_2.csv')['shipment_id'])
bayes_test_predictions['shipping_time'] = bayes_test_predictions

# print the df shape
print(bayes_test_predictions.shape)

# print the df head
```

```
bayes_test_predictions.head()
```

(1260, 2)

[165]:

	shipment_id	shipping_time
0	S002736	S002736
1	S002738	S002738
2	S005739	S005739
3	S008722	S008722
4	S009737	S009737

[166]:
```
# save the df as a csv file called submission_2.csv
bayes_test_predictions.to_csv('submission_2.csv', index=False)
```