Halloween Candies winpercent Prediction

- this dataset contain 85 diferent candy type and how often this candies were selected.
- `Done by :` Lujain Yousef.

Columns Explanation

- `competitorname:` 🍭
    - This column contains the names of the candies or competitors.
- `chocolate:` 🍫
    - This feature indicates whether the candy contains chocolate or not. A value of 1 typically means it contains chocolate, while 0 means it doesn't.
- `fruity:` 🍇
    - This feature indicates whether the candy has a fruity flavor. Again, 1 typically means it does, and 0 means it doesn't.
- `caramel:` 🍬
    - Indicates whether the candy contains caramel. 1 means it does, 0 means it doesn't.
- `peanutyalmondy:` 🥜

    - Indicates whether the candy contains peanuts or almonds. 1 means it does, 0 means it doesn't.
- `nougat:` 🍥
    - Indicates whether the candy contains nougat. 1 means it does, 0 means it doesn't.
- `crispedricewafer:` 🍪
    - Indicates whether the candy contains crisped rice or wafer. 1 means it does, 0 means it doesn't.
- `hard:` 🍩
    - Indicates whether the candy is hard. 1 means it is, 0 means it isn't.
- `bar:` 🍫
    - Indicates whether the candy is in bar form. 1 means it is, 0 means it isn't.
- `pluribus:` 🎁
    - Indicates whether the candy is part of a variety pack or if it's sold individually. 1 means it's part of a variety pack, 0 means it's sold individually.
- `sugarpercent:` 🧁
    - The percentage of sugar content in the candy.
- `pricepercent:` 💰
    - The relative price of the candy compared to others.
- `winpercent:` 🏆
    - The percentage of people who preferred this candy when compared to others in surveys or tests.

Reading The Dataset

In [1]:
```python
import pandas as pd
DF = pd.read_csv('candy-data.csv')
DF.head()
```

Out[1]:

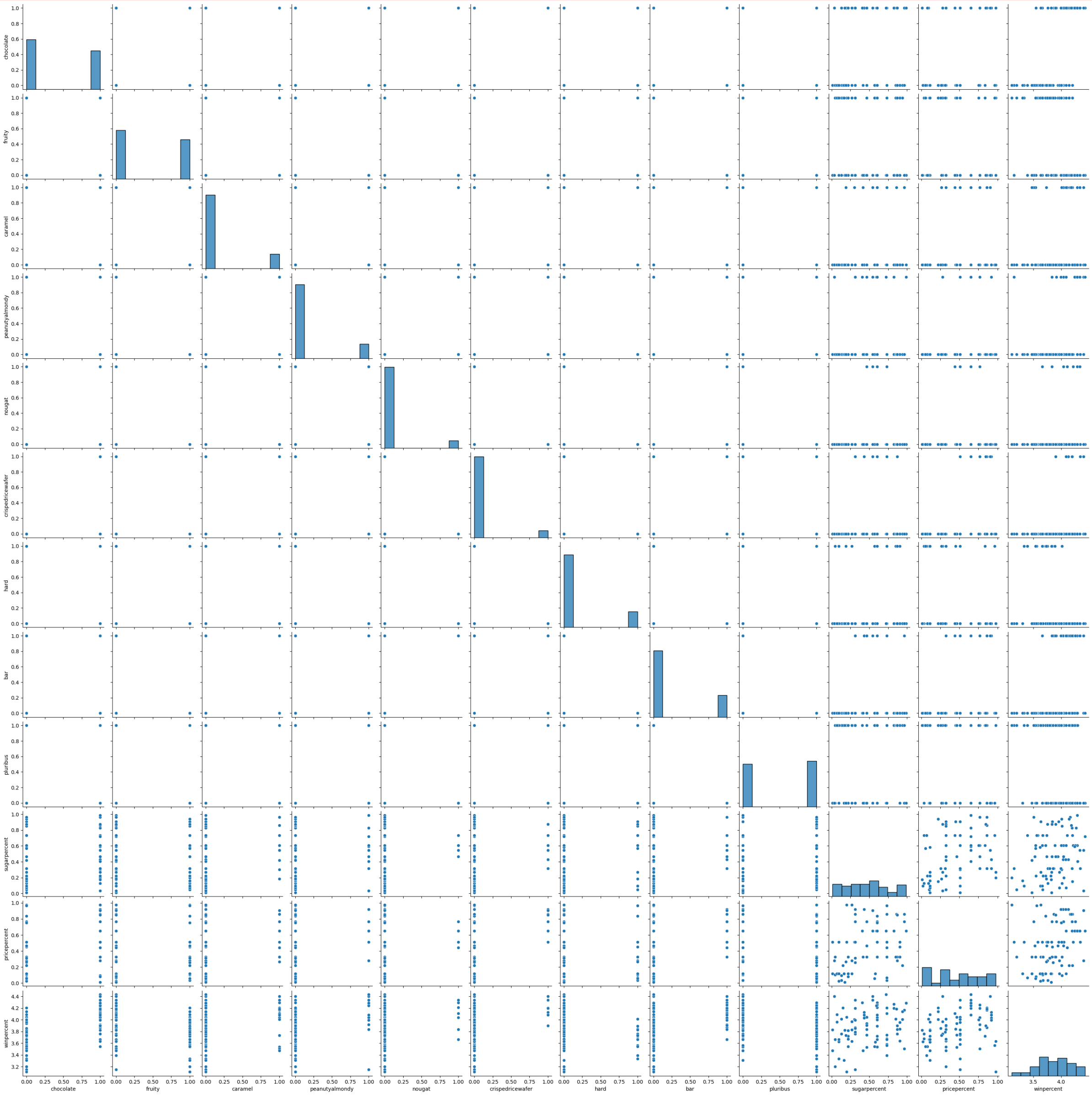
	competitorname	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard
0	100 Grand	1	0	1	0	0	1	0
1	3 Musketeers	1	0	0	0	1	0	0
2	One dime	0	0	0	0	0	0	0
3	One quarter	0	0	0	0	0	0	0
4	Air Heads	0	1	0	0	0	0	0

EDA

In [65]:

```
import seaborn as sns
sns.pairplot(DF);
```

```
C:\Users\yluja\Documents\adult.csv\Lib\site-packages\seaborn\axisgrid.py:118: User
Warning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [2]:

```
# Candies distinct types
DF['competitorname'].unique()
```

Out[2]:
```
array(['100 Grand', '3 Musketeers', 'One dime', 'One quarter',
       'Air Heads', 'Almond Joy', 'Baby Ruth', 'Boston Baked Beans',
       'Candy Corn', 'Caramel Apple Pops', 'Charleston Chew',
       'Chewey Lemonhead Fruit Mix', 'Chiclets', 'Dots', 'Dum Dums',
       'Fruit Chews', 'Fun Dip', 'Gobstopper', 'Haribo Gold Bears',
       'Haribo Happy Cola', 'Haribo Sour Bears', 'Haribo Twin Snakes',
       'HersheyÕs Kisses', 'HersheyÕs Krackel',
       'HersheyÕs Milk Chocolate', 'HersheyÕs Special Dark', 'Jawbusters',
       'Junior Mints', 'Kit Kat', 'Laffy Taffy', 'Lemonhead',
       'Lifesavers big ring gummies', 'Peanut butter M&MÕs', 'M&MÕs',
       'Mike & Ike', 'Milk Duds', 'Milky Way', 'Milky Way Midnight',
       'Milky Way Simply Caramel', 'Mounds', 'Mr Good Bar', 'Nerds',
       'Nestle Butterfinger', 'Nestle Crunch', 'Nik L Nip', 'Now & Later',
       'Payday', 'Peanut M&Ms', 'Pixie Sticks', 'Pop Rocks', 'Red vines',
       'ReeseÕs Miniatures', 'ReeseÕs Peanut Butter cup',
       'ReeseÕs pieces', 'ReeseÕs stuffed with pieces', 'Ring pop',
       'Rolo', 'Root Beer Barrels', 'Runts', 'Sixlets',
       'Skittles original', 'Skittles wildberry', 'Nestle Smarties',
       'Smarties candy', 'Snickers', 'Snickers Crisper',
       'Sour Patch Kids', 'Sour Patch Tricksters', 'Starburst',
       'Strawberry bon bons', 'Sugar Babies', 'Sugar Daddy',
       'Super Bubble', 'Swedish Fish', 'Tootsie Pop',
       'Tootsie Roll Juniors', 'Tootsie Roll Midgies',
       'Tootsie Roll Snack Bars', 'Trolli Sour Bites', 'Twix',
       'Twizzlers', 'Warheads', 'WelchÕs Fruit Snacks',
       'WertherÕs Original Caramel', 'Whoppers'], dtype=object)
```

In [4]:
```
DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85 entries, 0 to 84
Data columns (total 97 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   chocolate                           85 non-null     int64  
 1   fruity                              85 non-null     int64  
 2   caramel                             85 non-null     int64  
 3   peanutyalmondy                      85 non-null     int64  
 4   nougat                              85 non-null     int64  
 5   crispedricewafer                    85 non-null     int64  
 6   hard                                85 non-null     int64  
 7   bar                                 85 non-null     int64  
 8   pluribus                            85 non-null     int64  
 9   sugarpercent                        85 non-null     float64
 10  pricepercent                        85 non-null     float64
 11  winpercent                          85 non-null     float64
 12  name__100 Grand                     85 non-null     bool   
 13  name__3 Musketeers                  85 non-null     bool   
 14  name__Air Heads                     85 non-null     bool   
 15  name__Almond Joy                    85 non-null     bool   
 16  name__Baby Ruth                     85 non-null     bool   
 17  name__Boston Baked Beans            85 non-null     bool   
 18  name__Candy Corn                    85 non-null     bool   
 19  name__Caramel Apple Pops            85 non-null     bool   
 20  name__Charleston Chew               85 non-null     bool   
 21  name__Chewey Lemonhead Fruit Mix    85 non-null     bool   
 22  name__Chiclets                      85 non-null     bool   
 23  name__Dots                          85 non-null     bool   
 24  name__Dum Dums                      85 non-null     bool   
 25  name__Fruit Chews                   85 non-null     bool   
 26  name__Fun Dip                       85 non-null     bool   
 27  name__Gobstopper                    85 non-null     bool   
 28  name__Haribo Gold Bears             85 non-null     bool   
 29  name__Haribo Happy Cola             85 non-null     bool   
 30  name__Haribo Sour Bears             85 non-null     bool   
 31  name__Haribo Twin Snakes            85 non-null     bool   
 32  name__HersheyÕs Kisses              85 non-null     bool   
 33  name__HersheyÕs Krackel             85 non-null     bool   
 34  name__HersheyÕs Milk Chocolate      85 non-null     bool   
 35  name__HersheyÕs Special Dark        85 non-null     bool   
 36  name__Jawbusters                    85 non-null     bool   
 37  name__Junior Mints                  85 non-null     bool   
 38  name__Kit Kat                       85 non-null     bool   
 39  name__Laffy Taffy                   85 non-null     bool   
 40  name__Lemonhead                     85 non-null     bool   
 41  name__Lifesavers big ring gummies   85 non-null     bool   
 42  name__M&MÕs                         85 non-null     bool   
 43  name__Mike & Ike                    85 non-null     bool   
 44  name__Milk Duds                     85 non-null     bool   
 45  name__Milky Way                     85 non-null     bool   
 46  name__Milky Way Midnight            85 non-null     bool   
 47  name__Milky Way Simply Caramel      85 non-null     bool   
 48  name__Mounds                        85 non-null     bool   
 49  name__Mr Good Bar                   85 non-null     bool   
 50  name__Nerds                         85 non-null     bool   
 51  name__Nestle Butterfinger           85 non-null     bool   
 52  name__Nestle Crunch                 85 non-null     bool   
 53  name__Nestle Smarties               85 non-null     bool   
 54  name__Nik L Nip                     85 non-null     bool   
 55  name__Now & Later                   85 non-null     bool   
 56  name__One dime                      85 non-null     bool   
 57  name__One quarter                   85 non-null     bool   
 58  name__Payday                        85 non-null     bool   
```

```
 59  name__Peanut M&Ms                   85 non-null     bool
 60  name__Peanut butter M&MÕs           85 non-null     bool
 61  name__Pixie Sticks                  85 non-null     bool
 62  name__Pop Rocks                     85 non-null     bool
 63  name__Red vines                     85 non-null     bool
 64  name__ReeseÕs Miniatures            85 non-null     bool
 65  name__ReeseÕs Peanut Butter cup     85 non-null     bool
 66  name__ReeseÕs pieces                85 non-null     bool
 67  name__ReeseÕs stuffed with pieces   85 non-null     bool
 68  name__Ring pop                      85 non-null     bool
 69  name__Rolo                          85 non-null     bool
 70  name__Root Beer Barrels             85 non-null     bool
 71  name__Runts                         85 non-null     bool
 72  name__Sixlets                       85 non-null     bool
 73  name__Skittles original             85 non-null     bool
 74  name__Skittles wildberry            85 non-null     bool
 75  name__Smarties candy                85 non-null     bool
 76  name__Snickers                      85 non-null     bool
 77  name__Snickers Crisper              85 non-null     bool
 78  name__Sour Patch Kids               85 non-null     bool
 79  name__Sour Patch Tricksters         85 non-null     bool
 80  name__Starburst                     85 non-null     bool
 81  name__Strawberry bon bons           85 non-null     bool
 82  name__Sugar Babies                  85 non-null     bool
 83  name__Sugar Daddy                   85 non-null     bool
 84  name__Super Bubble                  85 non-null     bool
 85  name__Swedish Fish                  85 non-null     bool
 86  name__Tootsie Pop                   85 non-null     bool
 87  name__Tootsie Roll Juniors          85 non-null     bool
 88  name__Tootsie Roll Midgies          85 non-null     bool
 89  name__Tootsie Roll Snack Bars       85 non-null     bool
 90  name__Trolli Sour Bites             85 non-null     bool
 91  name__Twix                          85 non-null     bool
 92  name__Twizzlers                     85 non-null     bool
 93  name__Warheads                      85 non-null     bool
 94  name__WelchÕs Fruit Snacks          85 non-null     bool
 95  name__WertherÕs Original Caramel    85 non-null     bool
 96  name__Whoppers                      85 non-null     bool
dtypes: bool(85), float64(3), int64(9)
memory usage: 15.2 KB
```

In [57]:
```
DF.describe()
```

Out[57]:

	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard
count	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000
mean	0.435294	0.447059	0.164706	0.164706	0.082353	0.082353	0.176471
std	0.498738	0.500140	0.373116	0.373116	0.276533	0.276533	0.383482
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Standarization

In [63]:
```
DF.drop(columns='competitorname').var()
```

Out[63]:
```
chocolate             0.248739
fruity                0.250140
caramel               0.139216
peanutyalmondy        0.139216
nougat                0.076471
crispedricewafer      0.076471
hard                  0.147059
bar                   0.188235
pluribus              0.252661
sugarpercent          0.079963
pricepercent          0.081647
winpercent          216.512314
dtype: float64
```

`winpercent` has high var so i applied standarization

In [64]:
```
# log normalization to high var
import numpy as np
DF['winpercent'] = np.log(DF['winpercent'])
```

In [20]:
```
DF.drop(columns='competitorname').var()
```

Out[20]:

```
chocolate           0.248739
fruity              0.250140
caramel             0.139216
peanutyalmondy      0.139216
nougat              0.076471
crispedricewafer    0.076471
hard                0.147059
bar                 0.188235
pluribus            0.252661
sugarpercent        0.079963
pricepercent        0.081647
winpercent          0.091753
dtype: float64
```

One Hot Encoding

`competitorname` col has obj dtype so i applies One hot encoding

In [25]:

```
# Convert the Country column to a one hot encoded Data Frame
DF = pd.get_dummies(DF,columns=['competitorname'],prefix='name_')
DF.head()
```

Out[25]:

	chocolate	fruity	caramel	peanutyalmondy	nougat	crispedricewafer	hard	bar	pluribus	su
0	1	0	1	0	0	1	0	1	0	
1	1	0	0	0	1	0	0	1	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	1	0	0	0	0	0	0	0	

5 rows × 97 columns

Splitting the Data

In [27]:

```
# Dependent & independent Variables
X = DF.drop(['winpercent'],axis=1)
y = DF['winpercent']
```

In [28]:

```
# splitting
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
```

Hyperparameters tuning & Bulding the Model

Random Forest Regressor

In [51]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(random_state=1111)
```

In [54]:
```
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter grid for RandomizedSearchCV
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Create RandomizedSearchCV object
randomized_rf = RandomizedSearchCV(
    estimator=rfr,
    param_distributions=param_grid,
    n_iter=10,
    scoring='neg_mean_squared_error', #neg_mean_squared_error
    n_jobs=4,
    cv=10,
    refit=True,
    return_train_score=True
)

# Fit the RandomizedSearchCV object to your data
randomized_rf.fit(X_train, y_train)

# Get the best parameters
best_params = randomized_rf.best_params_

# Get the best estimator (model)
best_model = randomized_rf.best_estimator_

print('The best model is: \n', best_model)
print('Also, the best parameters are: \n', best_params)
```

```
The best model is: 
 RandomForestRegressor(min_samples_split=5, random_state=1111)
Also, the best parameters are: 
 {'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth':
None, 'bootstrap': True}
```

Most important features to the model

In [31]:
```
# Print how important each column is to the model
for i, item in enumerate(rfr.feature_importances_):
      # Use i and item to print out the feature importance of each column
    print("{0:s}: {1:.2f}".format(X_train.columns[i], item))
```

```
chocolate: 0.38
fruity: 0.04
caramel: 0.03
peanutyalmondy: 0.10
nougat: 0.00
crispedricewafer: 0.00
hard: 0.01
bar: 0.01
pluribus: 0.01
sugarpercent: 0.16
pricepercent: 0.09
name__100 Grand: 0.00
name__3 Musketeers: 0.00
name__Air Heads: 0.00
name__Almond Joy: 0.00
name__Baby Ruth: 0.00
name__Boston Baked Beans: 0.02
name__Candy Corn: 0.00
name__Caramel Apple Pops: 0.00
name__Charleston Chew: 0.00
name__Chewey Lemonhead Fruit Mix: 0.00
name__Chiclets: 0.02
name__Dots: 0.00
name__Dum Dums: 0.00
name__Fruit Chews: 0.00
name__Fun Dip: 0.00
name__Gobstopper: 0.00
name__Haribo Gold Bears: 0.00
name__Haribo Happy Cola: 0.00
name__Haribo Sour Bears: 0.01
name__Haribo Twin Snakes: 0.00
name__HersheyÕs Kisses: 0.00
name__HersheyÕs Krackel: 0.00
name__HersheyÕs Milk Chocolate: 0.00
name__HersheyÕs Special Dark: 0.00
name__Jawbusters: 0.01
name__Junior Mints: 0.00
name__Kit Kat: 0.00
name__Laffy Taffy: 0.00
name__Lemonhead: 0.00
name__Lifesavers big ring gummies: 0.01
name__M&MÕs: 0.00
name__Mike & Ike: 0.00
name__Milk Duds: 0.00
name__Milky Way: 0.00
name__Milky Way Midnight: 0.00
name__Milky Way Simply Caramel: 0.00
name__Mounds: 0.00
name__Mr Good Bar: 0.00
name__Nerds: 0.00
name__Nestle Butterfinger: 0.00
name__Nestle Crunch: 0.00
name__Nestle Smarties: 0.01
name__Nik L Nip: 0.01
name__Now & Later: 0.00
name__One dime: 0.00
name__One quarter: 0.00
name__Payday: 0.00
name__Peanut M&Ms: 0.00
name__Peanut butter M&MÕs: 0.00
name__Pixie Sticks: 0.00
name__Pop Rocks: 0.00
name__Red vines: 0.00
name__ReeseÕs Miniatures: 0.00
```

```
name__ReeseÕs Peanut Butter cup: 0.00
name__ReeseÕs pieces: 0.00
name__ReeseÕs stuffed with pieces: 0.00
name__Ring pop: 0.00
name__Rolo: 0.00
name__Root Beer Barrels: 0.00
name__Runts: 0.00
name__Sixlets: 0.00
name__Skittles original: 0.01
name__Skittles wildberry: 0.00
name__Smarties candy: 0.00
name__Snickers: 0.00
name__Snickers Crisper: 0.00
name__Sour Patch Kids: 0.00
name__Sour Patch Tricksters: 0.00
name__Starburst: 0.00
name__Strawberry bon bons: 0.00
name__Sugar Babies: 0.00
name__Sugar Daddy: 0.00
name__Super Bubble: 0.02
name__Swedish Fish: 0.01
name__Tootsie Pop: 0.00
name__Tootsie Roll Juniors: 0.00
name__Tootsie Roll Midgies: 0.00
name__Tootsie Roll Snack Bars: 0.00
name__Trolli Sour Bites: 0.00
name__Twix: 0.00
name__Twizzlers: 0.00
name__Warheads: 0.00
name__WelchÕs Fruit Snacks: 0.00
name__WertherÕs Original Caramel: 0.00
name__Whoppers: 0.00
```

`chocolate` is the most important feature

Model Evaluation

In [53]:
```
best_mae = -randomized_rf.best_score_
print("Best Mean Absolute Error:", best_mae)
```

```
Best Mean Absolute Error: 0.1708601783947124
```

In [55]:
```
best_mse = -randomized_rf.best_score_
print("Best Mean Squared Error:", best_mse)
```

```
Best Mean Squared Error: 0.0512847441934323
```

Now GradientBoosting Model

In [34]:
```
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor()
```

In [49]:
```
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
# Create RandomizedSearchCV object
randomized_gbr = RandomizedSearchCV(
    estimator=gbr,
    param_distributions=param_grid,
    n_iter=10,
    scoring='neg_mean_absolute_error', #neg_mean_squared_error
    n_jobs=4,
    cv=10,
    refit=True,
    return_train_score=True
)

# Fit the RandomizedSearchCV object to your data
randomized_gbr.fit(X_train, y_train)  # Assuming X_train and y_train are your trair

# Get the best parameters
best_params = randomized_gbr.best_params_
```

```python
# Get the best estimator (model)
best_model = randomized_gbr.best_estimator_
print('The best model is: \n', best_model)
print('Also, the best parameters are: \n', best_params)
```

```
The best model is:
 GradientBoostingRegressor(learning_rate=0.2, max_depth=5, min_samples_split=10)
Also, the best parameters are:
 {'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_dept
h': 5, 'learning_rate': 0.2}
```

Model Evaluation

In [50]:
```python
best_mae = -randomized_gbr.best_score_
print("Best Mean Absolute Error:", best_mae)
```

```
Best Mean Absolute Error: 0.1700472332797549
```

In [48]:
```python
best_mse = -randomized_gbr.best_score_
print("Best Mean Squared Error:", best_mse)
```

```
Best Mean Squared Error: 0.04886912431960155
```

XGBoost

In [44]:
```python
from sklearn.model_selection import RandomizedSearchCV
from xgboost import XGBRegressor
```

```python
# Define your XGBRegressor model
xgb = XGBRegressor()

# Define the parameter grid for RandomizedSearchCV
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 7, 10],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2, 0.3, 0.4],
    'reg_alpha': [0, 0.1, 0.5, 1.0],
    'reg_lambda': [0, 0.1, 0.5, 1.0],
}

# Create RandomizedSearchCV object
randomized_xgb = RandomizedSearchCV(
    estimator=xgb,
    param_distributions=param_grid,
    n_iter=10,
    scoring='neg_mean_squared_error',
    n_jobs=4,
    cv=10,
    refit=True,
    return_train_score=True
)

# Fit the RandomizedSearchCV object to your data
randomized_xgb.fit(X_train, y_train)

# Get the best parameters
best_params = randomized_xgb.best_params_

# Get the best estimator (model)
best_model = randomized_xgb.best_estimator_

print('The best model is: \n', best_model)
print('Also, the best parameters are: \n', best_params)
```

```
The best model is:
 XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.6, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=0, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.05, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=5, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=200, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
Also, the best parameters are:
 {'subsample': 1.0, 'reg_lambda': 0, 'reg_alpha': 0, 'n_estimators': 200, 'min_chi
ld_weight': 5, 'max_depth': 5, 'learning_rate': 0.05, 'gamma': 0, 'colsample_bytre
e': 0.6}
```

Model Evaluation

In [39]:

```python
best_mae = -randomized_xgb.best_score_
print("Best Mean Absolute Error:", best_mae)
```

```
Best Mean Absolute Error: 0.1856723316592046
```

In [45]:
```
best_mse = -randomized_xgb.best_score_
print("Best Mean Squared Error:", best_mse)
```

```
Best Mean Squared Error: 0.0587444232473864335
```