The Dataset came from Fast Food Nutrients from Kaggle. It contains 1,148 rows with 14 columns. The goal is to use Bayesian Linear Regression to predict the calories vs Linear Regression, RandomForest and XGBoost.

Please see jupyter notebook for full EDA and modeling

Percentage of 0s in the Data set

```
company                  0.00
item                     0.00
calories                 7.23
calories_from_fat       15.24
total_fat_(g)           31.10
saturated_fat_(g)       33.36
trans_fat_(g)           83.10
cholesterol_(mg)        32.93
sodium_(mg)              4.70
carbs_(g)                6.53
fiber_(g)               48.00
sugars_(g)              16.55
protein_(g)             27.35
weight_watchers_pnts     5.84
dtype: float64
```

Percentage of NaNs in the Data set

```
company                  0.00
item                     0.00
calories                 1.31
calories_from_fat       45.12
total_fat_(g)            6.01
saturated_fat_(g)        6.10
trans_fat_(g)            6.01
cholesterol_(mg)         2.53
sodium_(mg)              1.39
carbs_(g)                6.10
fiber_(g)                7.32
sugars_(g)               2.61
protein_(g)              6.01
weight_watchers_pnts    23.69
dtype: float64
```

Mean

```
Mean of the df
calories 287.36
total_fat_(g) 11.68
saturated_fat_(g) 4.07
cholesterol_(mg) 40.65
sodium_(mg) 427.72
carbs_(g) 38.95
sugars_(g) 24.07
protein_(g) 9.43
```

Median

```
Median of the df
calories 240.0
total_fat_(g) 8.0
saturated_fat_(g) 3.0
cholesterol_(mg) 20.0
sodium_(mg) 190.0
carbs_(g) 34.0
sugars_(g) 8.0
protein_(g) 7.0
```

The data transformation process involves several steps: fixing column names, converting columns from object types to floats, and dropping unnecessary columns.

In addition, distinguishing actual zero values from NaN values, as not all foods contain certain nutrients. For NaN values, we choose to use the median instead of the mean, as the mean tends to be higher (as observed in the EDA notebook).

To avoid concentrating data in one unknown value, we didn’t use forward or backward fill methods also, it would be too much computation heavy that would drag the program if we try to impute individually. Additionally, we drop the columns ['trans_fat_(g)', 'fiber_(g)', 'calories_from_fat', 'weight_watchers_pnts'] due to huge amount of 0s or NaNs.

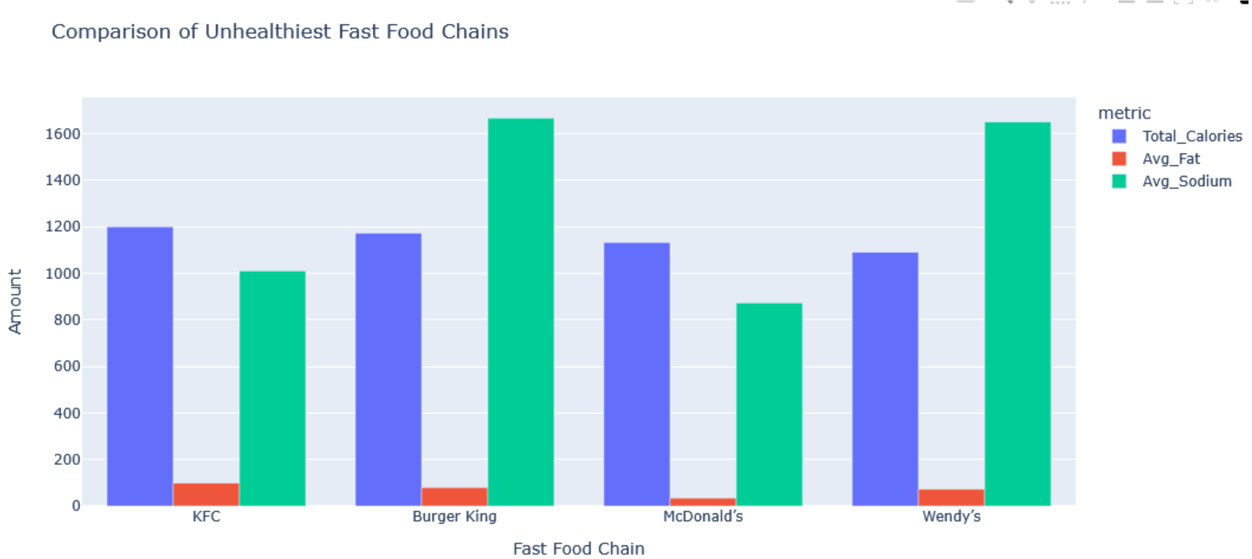
Fast Food “Nutrients” vs Daily Limit by consuming single item an average person already hit ‘nutrition limit’

Nutrient	Daily Limit	Worst Offender (Item & Company)	Value	FDA Daily Limit	Estimated Per-Item Limit (1 meal = ~⅓ of daily intake)	Excess (Over/Under Limit)
Calories	2,000 kcal	Triple Whopper® w/ Cheese (Burger King)	1220 kcal (61% of daily limit)	2,000 kcal	~667 kcal per meal	553 kcal (Over)
Total Fat	70g	Potato Salad (Family) (KFC)	98g (140% over limit)	78g	~26g per meal	72g (Over)
Sodium	2,300mg	Secret Recipe Fries (Family) (KFC)	2890mg (125% over limit)	2,300mg	~767mg per meal	2123mg (Over)
Carbs	275g	Strawberry Lemonade (1/2 Gallon) (KFC)	270g (98% of limit)	275g	~92g per meal	178g (Under)
Protein	50g	Triple Whopper® w/ Cheese (Burger King)	71g (142% over limit)	50g	~17g per meal	54g (Over)

Company level insights (Highest per nutrient average)

Calories	Burger King: 359 kcal per item
Sodium content	Burger King: 540 mg (very high sodium levels)
Fat	Burger King: 16.6g per item
Highest Sugar Content (g)	McDonald’s: 28.1g per item (highest sugar levels)

Burger king is consistent offender



Nutrient Correlation

Nutrient Pair	**Correlation Strength**	**Key Insight**
Calories & Fat	Strong (0.83)	Higher fat = higher calories, but sugary items (sodas, shakes) can be exceptions.
Calories & Sodium	High (0.73)	High-calorie foods tend to be high in sodium, likely due to processed ingredients.
Fat & Sodium	Strong (0.81)	Fatty foods are often salt-heavy (e.g., fried chicken, burgers).
Calories & Sugars	Weak (0.26)	Sugary items (shakes, sodas) are calorie-dense but don't always have fat.
Fat & Sugars	Negative (-0.23)	High-fat foods (burgers) usually don't have much sugar, while high-sugar items (soft drinks) are often fat-free.
Calories & Protein	Moderate (0.73)	Higher protein often means more calories, but sources matter (grilled vs. fried chicken).

Features as predictors

From MC01 **Random Forest** we got the following;

Feature	**Coef**
Total Fat	0.65
Carbs	0.21
Sodium	0.046
Saturated Fat	0.042
Protein	0.023

For MC02 we utilized **PCA** to identify most important feature to use as predictor

Feature	**Loadings**
Total Fat	0.46
Protein	0.45
Saturated Fat	0.45
Sodium	0.44
Cholesterol	0.39

Features ranked by correlation with PCA1

Feature	Correlation
Protein	0.94
Total Fat	0.88
Cholesterol	0.82
Sugar	0.81
Saturated Fat	0.65
Sodium	0.18
Carbs	-0.23

Machine Learning Models

In this project, 3 Machine learning models (Linear Regression with L1 and Polynomial Features (degree =2), Random Forest and XGBoost) are used in predicting the calories given features. also utilized the best parameters from lasso to the polynomial LR.

The model has test size of 20%, cross validation =7 and GridSearchCV.

```
# Hyperparameters for Lasso LR
lasso_param_grid = {
    'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'fit_intercept': [True, False]
}

# Hyperparameters for RF
rf_param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True, False]
}

# Hyperparameters for XGBoost
xgb_param_grid = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 5, 7],
    'subsample': [0.7, 0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'alpha': [0, 0.1, 1]
}
```

With Best Parameters as;

Best Hyperparameters from Grid Search

- **Lasso Regression**
  - `alpha` : 0.1
  - `fit_intercept` : True
- **Random Forest Regressor**
  - `bootstrap` : False
  - `max_depth` : None
  - `max_features` : 'sqrt'
  - `min_samples_leaf` : 1
  - `min_samples_split` : 2
  - `n_estimators` : 200
- **XGBoost Regressor**
  - `alpha` : 0
  - `colsample_bytree` : 1.0
  - `gamma` : 0.2
  - `learning_rate` : 0.1
  - `max_depth` : 7
  - `n_estimators` : 100
  - `subsample` : 0.7

Bayesian LR with MCMC

The data used was also split with 20% test set with Kfolds = 5, for MCMC it was set to have 500 warmup and samples of 1000 in 4 chains.

Fold	**MAE**	**MSE**	**RMSE**	**R^2**
Fold 1	0.172780	0.061463	0.247916	0.944704
Fold 2	0.167734	0.062106	0.249212	0.917112
Fold 3	0.203737	0.098117	0.313237	0.906646
Fold 4	0.157474	0.055755	0.236124	0.937994
Fold 5	0.183060	0.069879	0.264345	0.929270

Results vs Train Set

Model	Train MSE	Train R^2
Lasso	3243.40	*0.9318*
Random Forest	25.21	*0.9995*
XGBoost	123.12	*0.9974*
Lasso with Polynomial Features	2266.99	*0.9590*
Bayesian Linear Regression	3589.26	*0.9352*

Results MC01 vs Test Set

Model	MAE	MSE	RMSE	R^2
Linear Regression	0.4849	0.0037	0.0605	0.8913
Linear Regression with MCMC	0.4525	0.0028	0.0532	0.9020

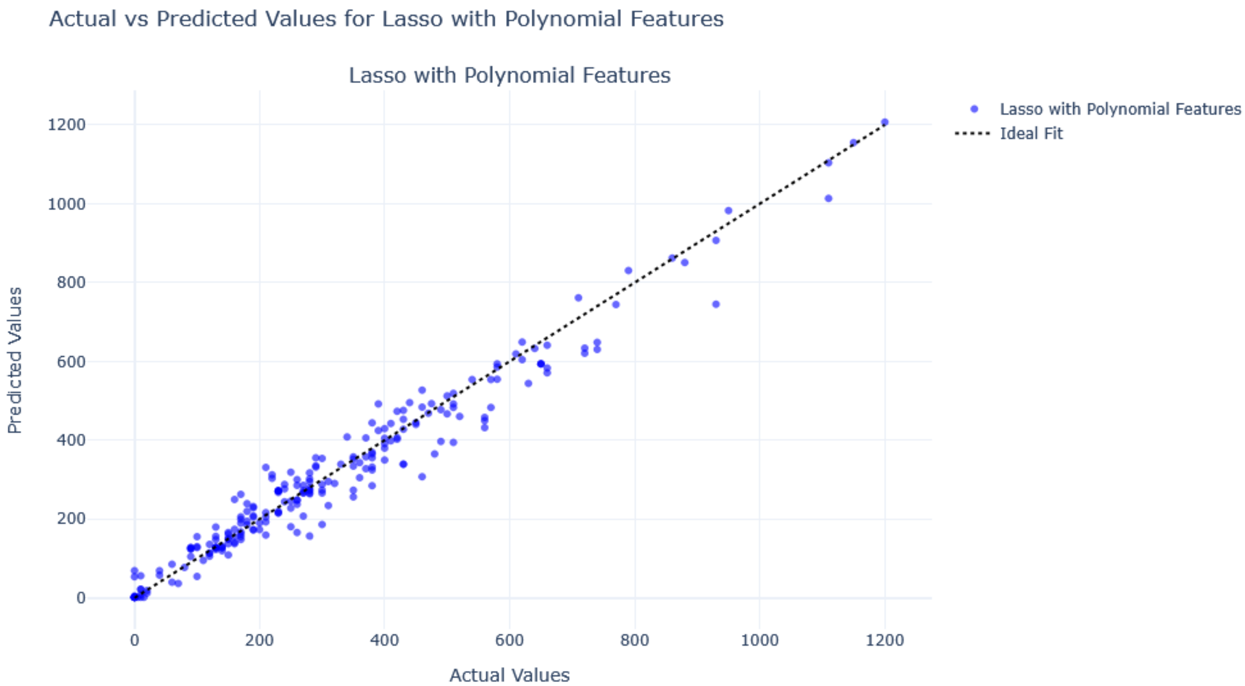
Results from MC02 Test Set

Model	MAE	MSE	RMSE	R^2
Lasso	42.0299	3574.1742	59.7843	0.9354
Random Forest	25.2086	2128.6516	46.1373	0.9615
XGBoost	22.9829	1481.5767	38.4912	***0.9732***
Lasso with Poly Features	33.9385	2266.9894	47.6129	0.9590
Bayesian LR W/ MCMC	42.1617	3589.2590	59.9104	0.9352

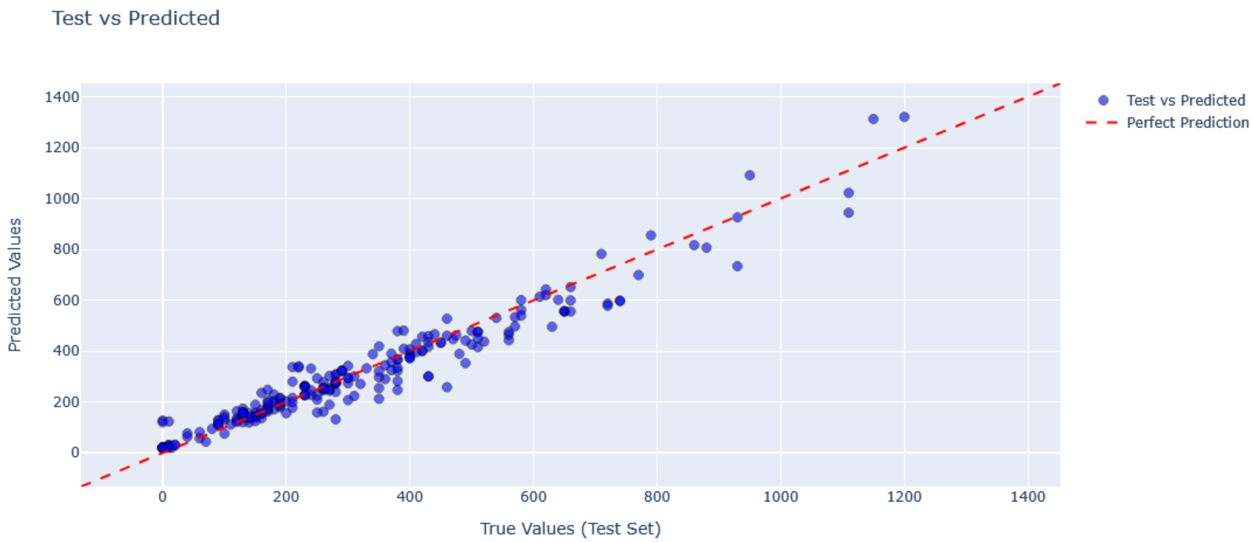
Linear Regression with Lasso, Poly features and Bayesian Model with Kfolds performed better than (MC01) the traditional Linear model and basic Bayesian model. While XGBoost performed the best with the lowest MAE, MSE, RMSE, and the highest R^2 with Lasso Regression and Bayesian the weakest performer.

Scatter Plot for ML Models

Polynomial LR Scatter Plot



Bayesian LR with MCMC



Models vs New Data

food	protein_(g)	total_fat_(g)	cholesterol_(mg)	sugars_(g)	saturated_fat_(g)	calories
Chicken Breast	31.0	3.6	85	0.0	1.0	165
Apple	0.5	0.3	0	19.0	0.0	52
Banana	1.3	0.4	0	14.0	0.1	89
Almonds	21.0	49.0	0	3.9	3.7	576
Avocado	2.0	21.0	0	0.2	3.1	160
Broccoli	3.7	0.4	0	1.7	0.1	55
Egg	6.0	5.0	186	0.6	1.6	68
Salmon	22.0	13.0	60	0.0	1.9	208
Greek Yogurt	10.0	0.0	10	6.6	0.0	59
Sweet Potato	2.0	0.2	0	4.2	0.0	86

Predictions on new data

Food	Actual Calories	Lasso Prediction	Lasso with Polynomial Prediction	Random Forest Prediction	XGBoost Prediction	Bayesian LR model
Chicken Breast	165	205.058514	207.390680	169.800	192.718948	204.882599
Apple	52	90.160559	77.090602	113.450	93.883789	90.000984
Banana	89	78.668043	69.315906	86.325	92.062569	78.475929
Almonds	576	777.251694	834.242281	701.525	860.208740	781.374695
Avocado	160	301.902949	336.137621	292.650	249.978683	302.991119
Broccoli	55	49.553088	50.153893	69.250	81.607689	49.317848
Egg	68	66.673030	-17.543801	73.050	13.708838	65.202705
Salmon	208	286.363281	301.288287	243.575	263.849518	286.905090
Greek Yogurt	59	91.312990	106.776963	125.800	118.015968	91.159172
Sweet Potato	86	46.556743	39.327799	49.675	53.703541	46.305397

Metrics

Model	MAE	MSE	RMSE	R^2
Random Forest	48.510000	4443.473750	66.659386	**0.803893**
Lasso Regression	58.859908	7257.848674	85.193008	0.679683
Bayesian LR	59.601178	7464.667689	86.398308	0.670556
XGBoost	67.491353	10270.335648	101.342665	0.546731
Lasso with Polynomial Features	79.967264	12103.237238	110.014714	0.465838

Results: based from Train, Test and New data

The Random Forest model shows a significant drop in performance when evaluated on new, unseen data, the model has R^2 from 0.9995 on the train set to 0.8039 on the actual test data. The model could be overfitted due to the large difference between the training and actual test.

The Lasso Regression model performs good on training and test data, with a slight difference in performance metrics. While, the performance on actual test data significantly worsens (underfitting), especially in terms of R^2, which drops to 0.6797.

Bayesian Linear Regression displays consistent performance on both training and test data, with a similar R^2 of 0.9352. The model significantly worsens on the actual test data (underfitting), with R^2 dropping to 0.6706.

XGBoost performs very well on the training data (R^2 of 0.9974) and shows good performance on the test data (R^2 of 0.9732). The model also experiences enormous decline on actual test data, with R^2 falling to 0.5467. The model captures both training and test set so much but fails to compare with the actual data suggesting overfitting.

The Lasso with Polynomial Features model shows good performance on both the training and test data, with very similar R^2 values of 0.9590. However, the performance on the actual test data is much worse, with R^2 dropping to 0.4658. This large difference indicates severe overfitting this may indicate that the model is too much that it likely learned to capture even noise specially on high degree (polynomial) features and it’s not good at capturing new data.

Back in MC01 Bayesian LR managed to outperform a bit (~1%) simple LR while on the new model the LR with L1 regularization manage to be on the same level with the Bayesian LR but both are outclassed by better models (Random Forest and XGBoost) due to its ‘nature’ of being tree-based algo which makes them less sensitive and are capable of capturing non-linear relationships.

Confidence Interval New Data

95% CI	**Lasso**	**Bayesian**	**Poly**	**RF**	**XG**
Mean Lower Bound	46.5567	46.9832	-17.5438	48.6667	13.7088
Mean Upper Bound	777.2517	673.7384	834.2423	676.8667	860.2087

From the 3 Linear Models, **Bayesian LR** seems to be the most stable and precise models, with narrower confidence intervals compared to Lasso and LR with Poly Features.

Random Forest Model shows narrow results along with Bayesian model.

LR with Poly features model looks to be problematic since it interprets it answer as negative which suggests that the model’s instability in handling the polynomial transformation of the data could be because of correlations with the dataset even though we performed L1 regularization or the actual models needs more work.

Lasso, Poly LR and XGboost predictions lean on uncertainty due to high variance, which supports our summary that these models are overfitting.