Campaign Ads Analysis Project

Optimizing Targeted Audience

## Goal:

The goal is to model an optimal audience-targeting strategy that maximizes viewable impressions

for advertisers.

It is also expected to model out an optimal bidding strategy and/or maximizing “new to brand” reach.

## Data

## **Dictionary:**

| **column** | **definition** |
| --- | --- |
| ADV\_ID | advertiser id |
| AD\_ID | ad id |
| SKU | item identifier |
| placement\_slot | where ad ran |
| impressions | number of times an ad ran (and was viewed) |
| clicks | number of clicks it generated |
| auction\_cost | Auction cost is how much it cost to win the auction |
| adjusted\_cost | adjusted cost should be how much they actually paid given it's a second price auction |
| hit\_day\_utc | day ad ran |
| targeting\_secondary | audience segment |
| vertical | advertiser "best guess" vertical by their account managers |
| sub\_vertical | advertiser "best guess" sub-vertical by their account managers |
| ntb\_click\_attributed\_units\_sold | new to brand (ntb) units sold that are attributable to clicks |
| ntb\_view\_attributed\_units\_sold | new to brand (ntb) units sold that are attributable to views |
| ntb\_click\_attributed\_orders | new to brand (ntb) orders that are attributable to clicks |
| ntb\_view\_attributed\_orders | new to brand (ntb) orders that are attributable to views |
| view\_attributed\_units\_sold | total units sold attributable to views |
| view\_attributed\_orders | total orders attributable to views |

#### 

#### **2. Feature Engineering:**

* Targeting Categories: One-hot encoding of targeting\_secondary
* Ad Placement: One-hot encoding of placement\_slot
* Temporal Features: Extract features like day of the week, month, etc., from hit\_day\_utc

#### **3. Model Selection:**

* Regression Models:
  + Gradient Boosting Regressor
  + Random Forest Regressor

We decided to go ahead with regression model to predict impressions and then use it to optimize audience targeting. ​

1. Mean Absolute Error (MAE):-​

* Definition: It tells you how far your predictions are from the actual values on average.​

​

1. Mean Squared Error (MSE):-​

* Definition: It shows how much your predictions differ from the actual values giving more weight to bigger mistakes.​

​

1. Evaluation Metric R-squared (R²):-​

* Definition: It tells you how well your model's predictions match the actual data, in a range between 0 to 1, with 1 being a perfect match and 0 indicating no match.​

R-squared (R2) is defined as a number that tells you how well the independent variable(s) in a model explain the variation in the dependent variable. It goes from 0 to 1, where 1 indicates a perfect fit of the model to the data.​

​

Random Forest Regressor: Think of a random forest as a group of independent graders. Each grader (decision tree) grades the homework independently and doesn't learn from the mistakes of others. At the end, you combine all the grades to get the final score by averaging them. This method helps to reduce the impact of any one grader's mistakes by using the collective wisdom of the group.​

​

Gradient Boosting Regressor: Imagine you have a team of graders who work sequentially. The first grader gives the initial grades. The second grader then focuses on correcting the mistakes made by the first grader. The third grader corrects the mistakes of the first two graders, and so on. This way, each grader builds on the work of the previous ones, progressively improving the accuracy of the final grade. This process is called "boosting."​

​

​

* Gradient Boosting:​
* Gradient Boosting builds trees sequentially, where each new tree corrects errors made by the previous ones.​
* It is an additive model, meaning it adds new models to correct errors made by existing models.​
* It typically uses shallow trees as base learners, often decision trees with a depth of 1 to 5.​
* Gradient Boosting can overfit if the number of trees is too large or if the learning rate is too high.​
* Random Forest:​
* Random Forest builds multiple decision trees independently and combines their predictions through averaging or voting.​
* It trains each tree on a random subset of the data (bootstrapped sample) and a random subset of the features.​
* It can handle a large number of features and is less prone to overfitting compared to Gradient Boosting.​
* Random Forest can be parallelized as each tree can be trained independently.

#### **4. Optimization:**

* Target Audience Segments: Use model predictions to identify high-impression audience segments.
* Bidding Strategy: Develop bidding strategies that align with the budget while maximizing impressions.

#### **5. Evaluation:**

* Regression Metrics: MAE, MSE, R-squared

We decided to go ahead with regression model to predict impressions and then use it to optimize audience targeting.

## Data Cleaning:

Data Cleaning: Combined the two datasets and deduplicated data based on repeated dates.

**Data Preparation**:

* Filled missing values for categorical features ('vertical', 'sub\_vertical') with 'Unknown'.
* Filled missing values in numerical features with 0.(last 6 columns)
* Parsed date information from 'hit\_day\_utc' to extract 'month' and 'day\_of\_week'.

## EDA:

**A graph showing a number of women&#39;s and men&#39;s sales

Description automatically generated**

# Models:

Green highlighted models are used in Final presentation

# Impressions

1. Whole dataset for modeling taking into account impressions, clicks, targeting\_secondary, placement\_slot, day of week, month, vertical, sub\_vertical. Blank vertical, sub\_vertical are considered Unknown and used in modeling. Target is impressions. Model: Gradient Boosting Rehressor

**No filtering, observed one whole dataset**

features\_cost = ['clicks', 'targeting\_secondary', 'placement\_slot', 'month\_name', 'day\_name', 'vertical', 'sub\_vertical']

target\_cost = 'impressions'

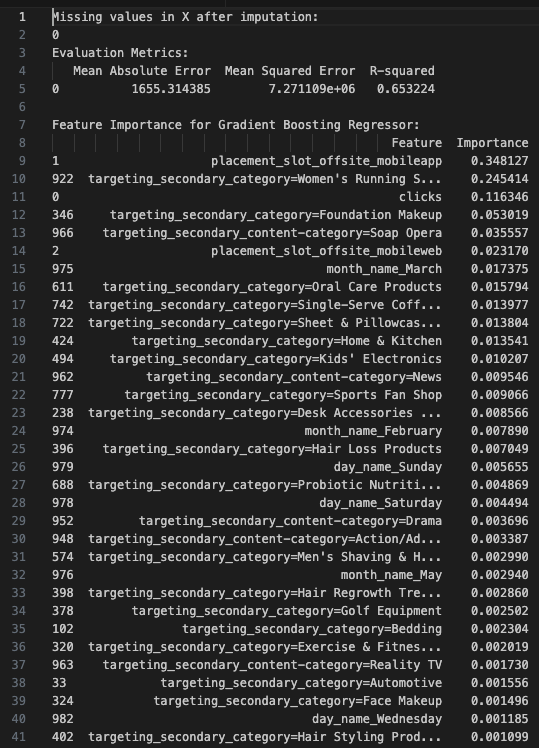


## Optimal Strategy for maximizing impressions:

1. Optimal modeling to maximize impressions where view\_attributed\_units\_sold(total units sold attributable to views) is atleast 1

Filtered dataset: view\_attributed\_units\_sold > 0

* Target: 'impressions’​
* Features: ‘clicks’, 'placement\_slot\_’, 'targeting\_secondary\_’, 'month\_name\_’, 'day\_name\_'



1. Random Forest regressor

Filtered dataset: view\_attributed\_units\_sold > 0

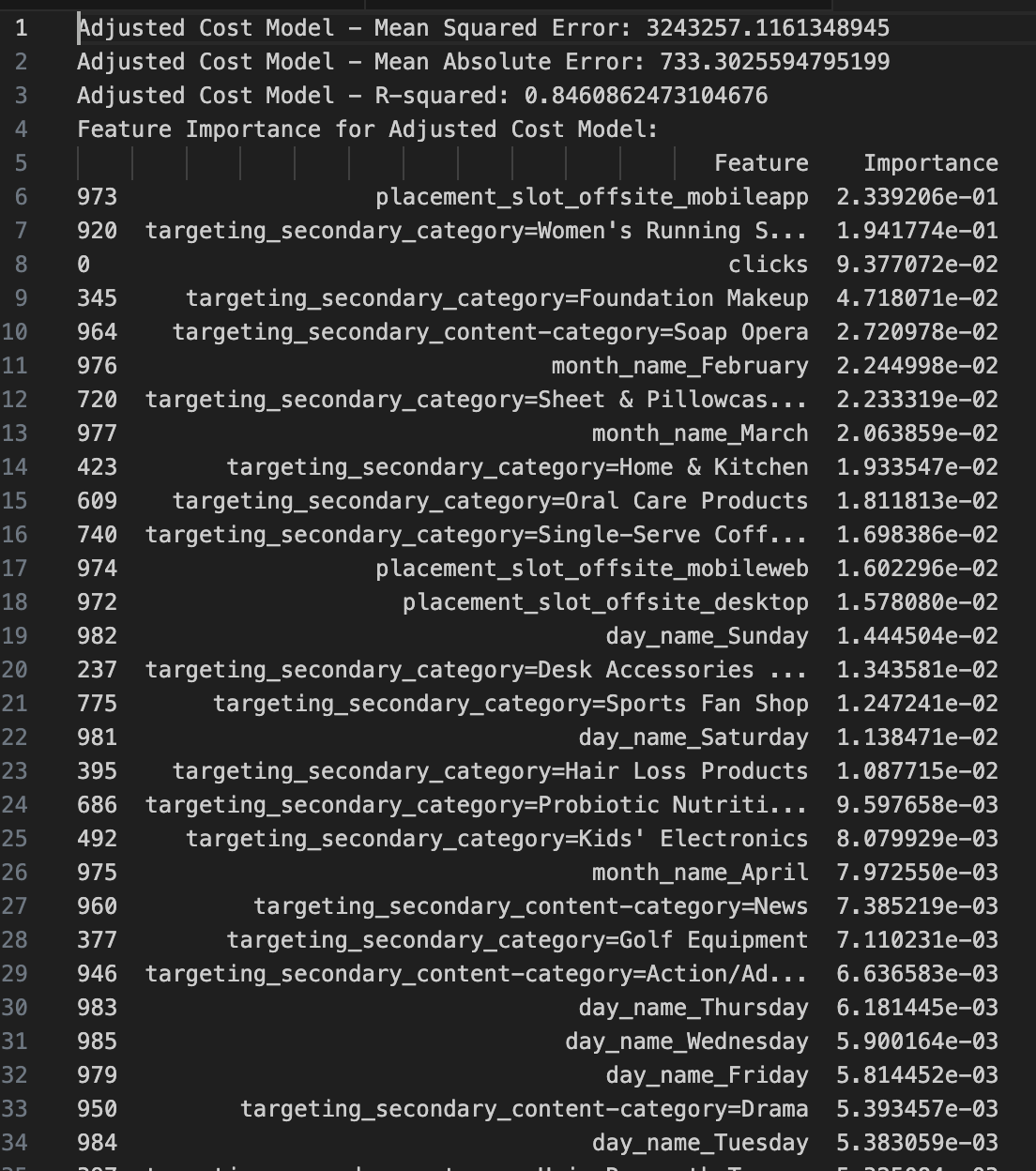
Target: 'impressions’​

Features: ‘clicks’, 'placement\_slot\_’, 'targeting\_secondary\_’, 'month\_name\_’, 'day\_name\_'

Adjusted Cost Model - Mean Squared Error: 3243257.1161348945

Adjusted Cost Model - Mean Absolute Error: 733.3025594795199

Adjusted Cost Model - R-squared: 0.8460862473104676



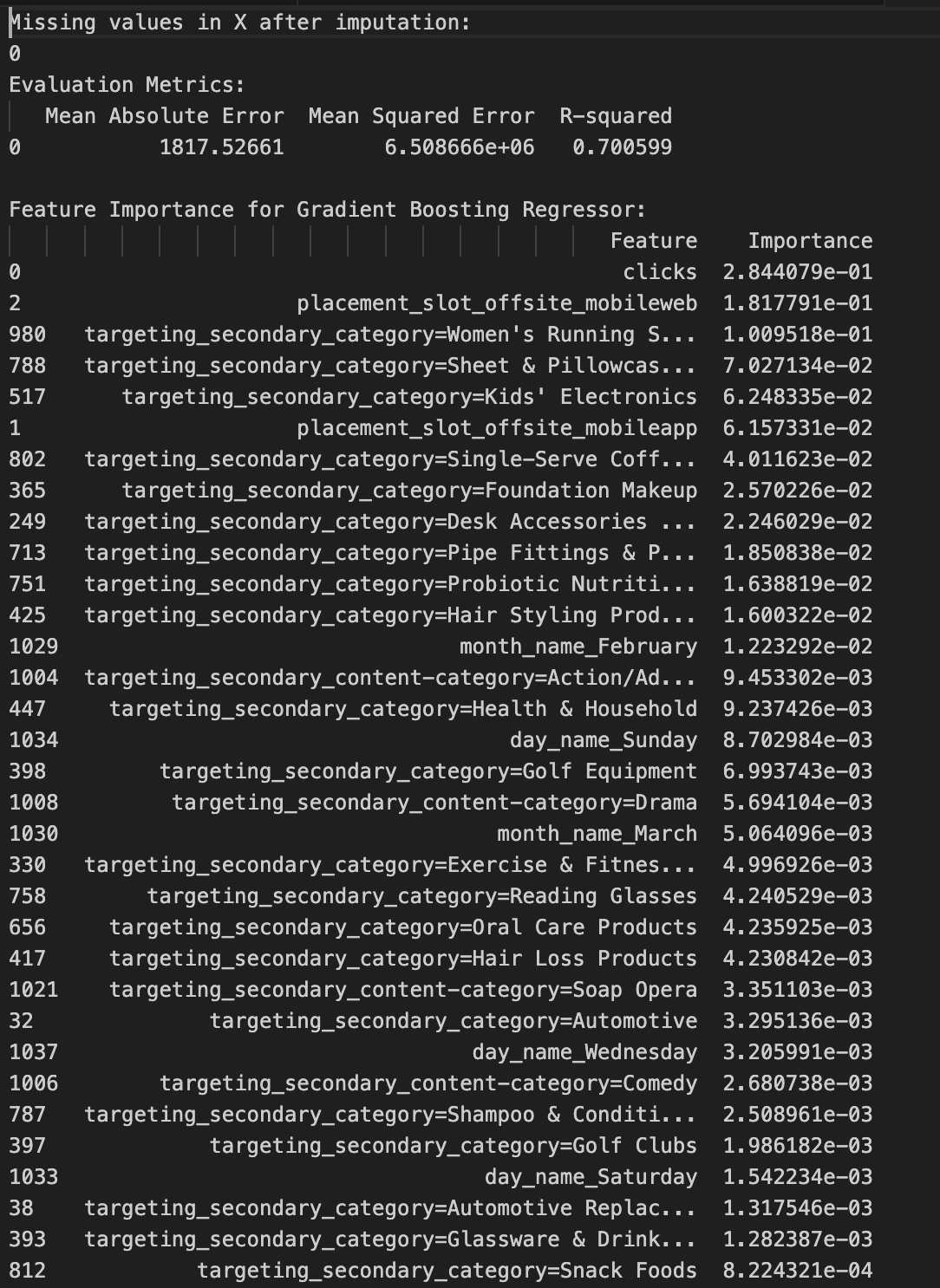
1. **Optimized Impressions with no verticals considering clicks in features(Click > 0)**

**Gradient boosting Model**

Evaluation Metrics:

Mean Absolute Error Mean Squared Error R-squared

0 1817.52661 6.508666e+06 0.700599



**4. RandomforestRegressor Model**

Subset: clicks > 0

* Target: impressions
* Features: clicks, 'placement\_slot', 'targeting\_secondary', 'month\_name', 'day\_name'

Mean Absolute Error Mean Squared Error R-squared

0 543.107426 1.746526e+06 0.922257

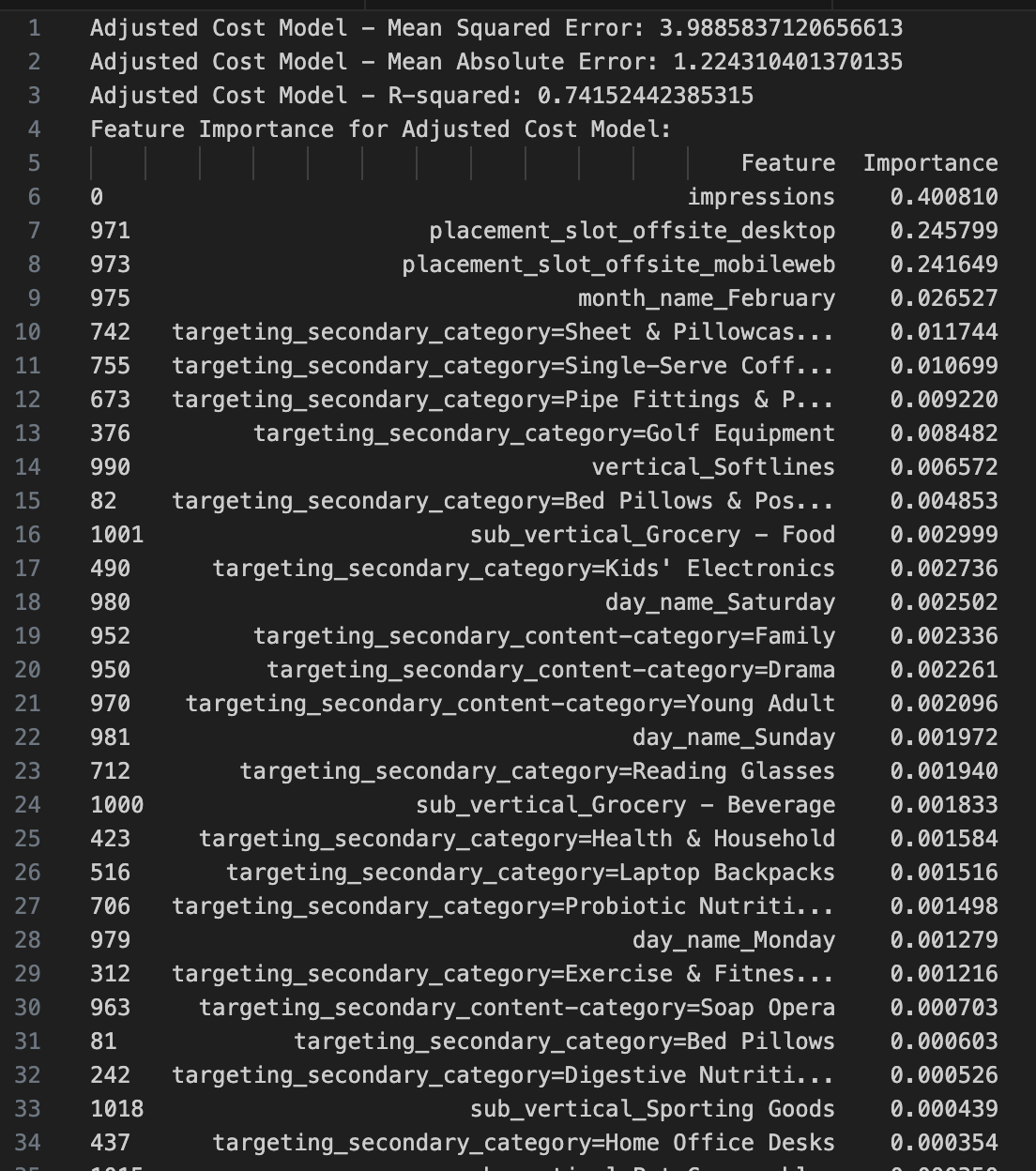


## Maximize CLICKS

**GradientBoostingRegressor Model**

* Target: ‘clicks’​
* Features: ‘impressions’, 'placement\_slot', 'targeting\_secondary', 'vertical', 'sub\_vertical', 'month\_name', 'day\_name'

Filtered data: Clicks atleast 1



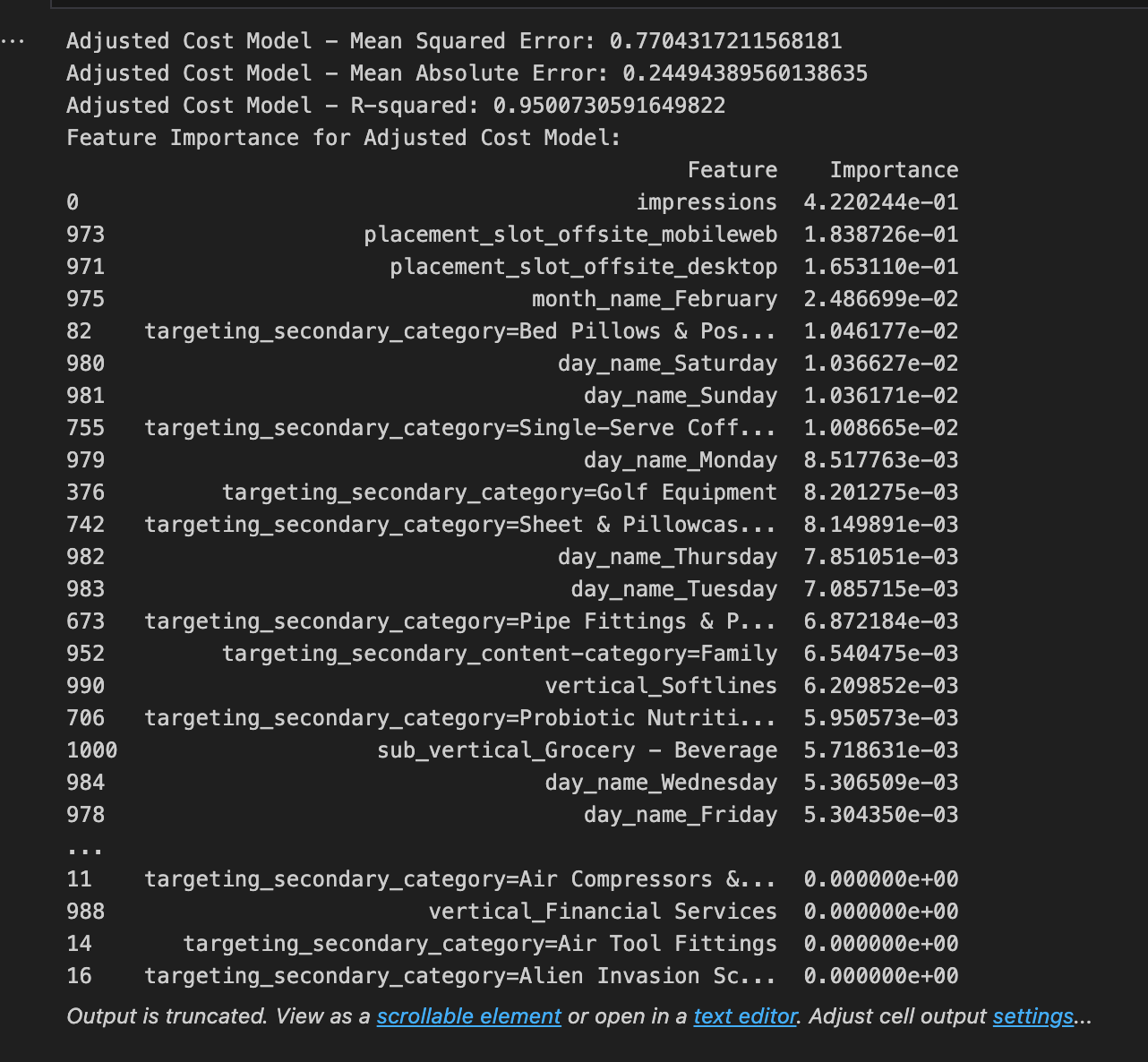
2. **RandomforestRegressor Model**

* Target: clicks
* Features:’impressions’, 'placement\_slot', 'targeting\_secondary', 'month\_name', 'day\_name'

Adjusted Cost Model - Mean Squared Error: 0.7704317211568181

Adjusted Cost Model - Mean Absolute Error: 0.24494389560138635

Adjusted Cost Model - R-squared: 0.9500730591649822



Both models give less errors so choosing gradient boosting as it’s more advanced technique.

## **NTB reach:**

We are considering units sold instead of orders, as sold is a surerity that item has been sold and not only placed as an order. Also if more units are being sold they could be popular and maximizing strategy for them will help to get more profit. We consider data where ntb\_view\_attributed\_units\_sold is not ‘0’ and not in ‘blank’.

We take view column of ntb to have maximum data while training and because we want to focus on ntb reach rather than only clicks. We have considered one cost as feature in final model and also seen that there isn’t much difference when two type of costs are considered.

1. **If we take impressions as target then below are the results: GradientBoostingRegressor**

Filtered data: ntb\_view\_attributed\_units\_sold is not ‘0’ and not in ‘blank’.

features\_ntb = ['targeting\_secondary', 'placement\_slot', 'month\_name', 'day\_name']

target\_ntb = 'impressions'

NTB Reach Model - Mean Squared Error: 8475900.0519299

NTB Reach Model - Mean Absolute Error: 1791.8868942504591

NTB Reach Model - R-squared: 0.613637876147813



1. If we take impressions as target then below are the results: RandomForestRegressor

Filtered data: ntb\_view\_attributed\_units\_sold is not ‘0’ and not in ‘blank’.

features\_ntb = ['targeting\_secondary', 'placement\_slot', 'month\_name', 'day\_name']

target\_ntb = 'impressions'



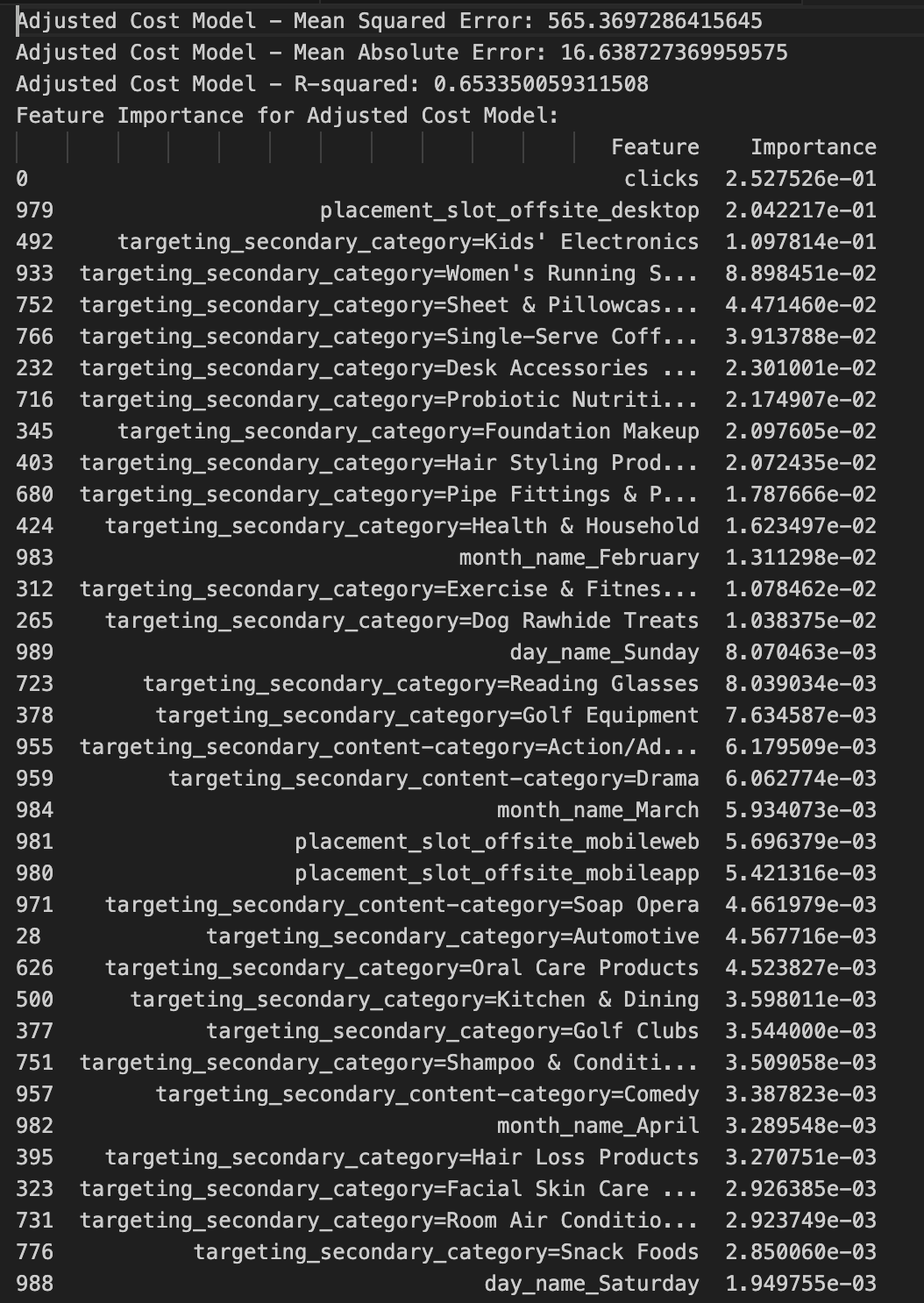
## Optimal bid (adjusted\_cost)

Taking adjusted cost as target instead of auction\_cost and subset where there has been some click.

**GradientBoostingRegressor**

features\_cost = [ 'targeting\_secondary', 'placement\_slot', 'month\_name', 'day\_name']

target\_cost = 'adjusted\_cost'

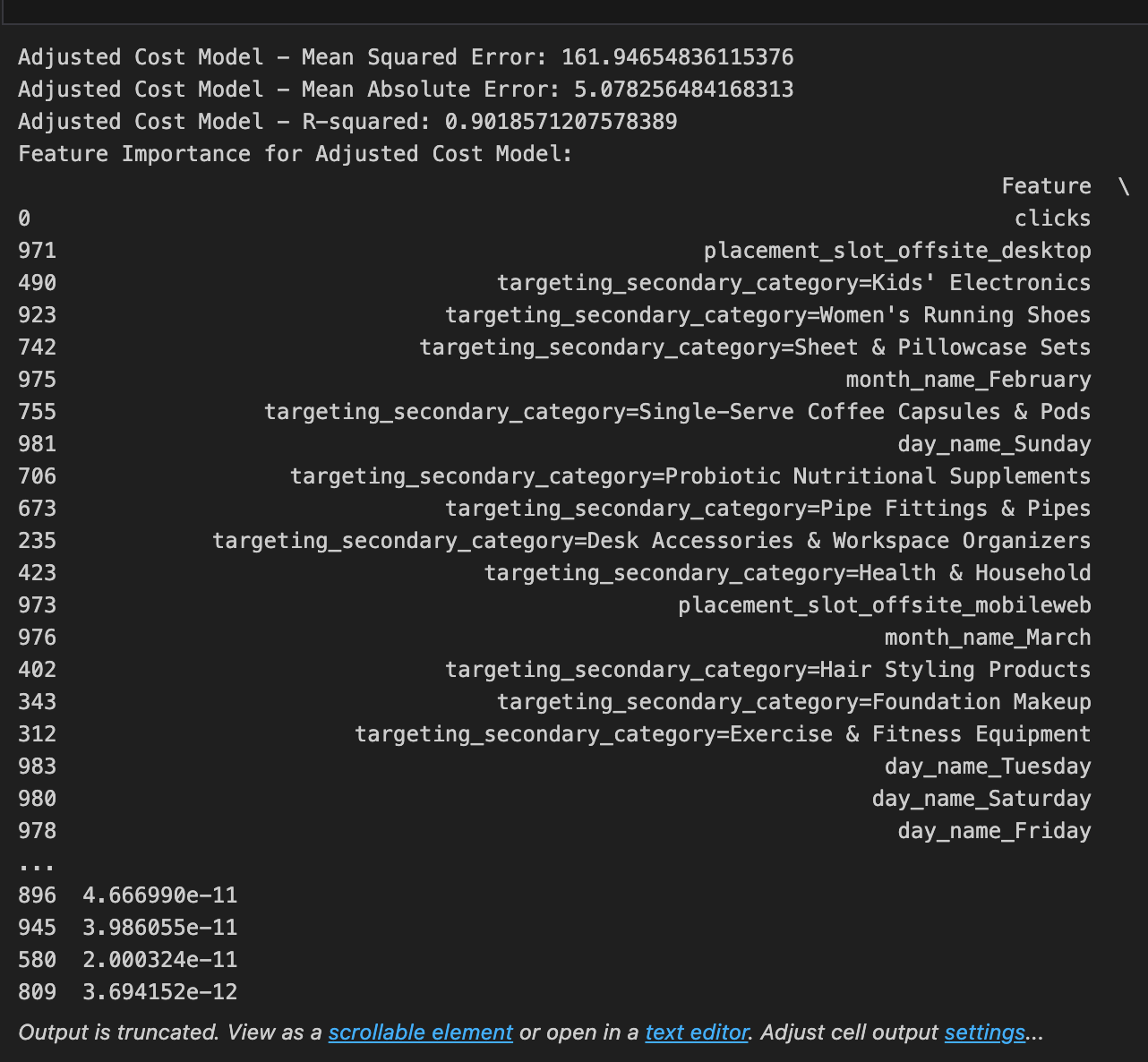
****

**RandomForestRegressor:**

subset where there has been some click.

features\_cost = [ 'clicks', 'targeting\_secondary', 'placement\_slot', 'month\_name', 'day\_name']

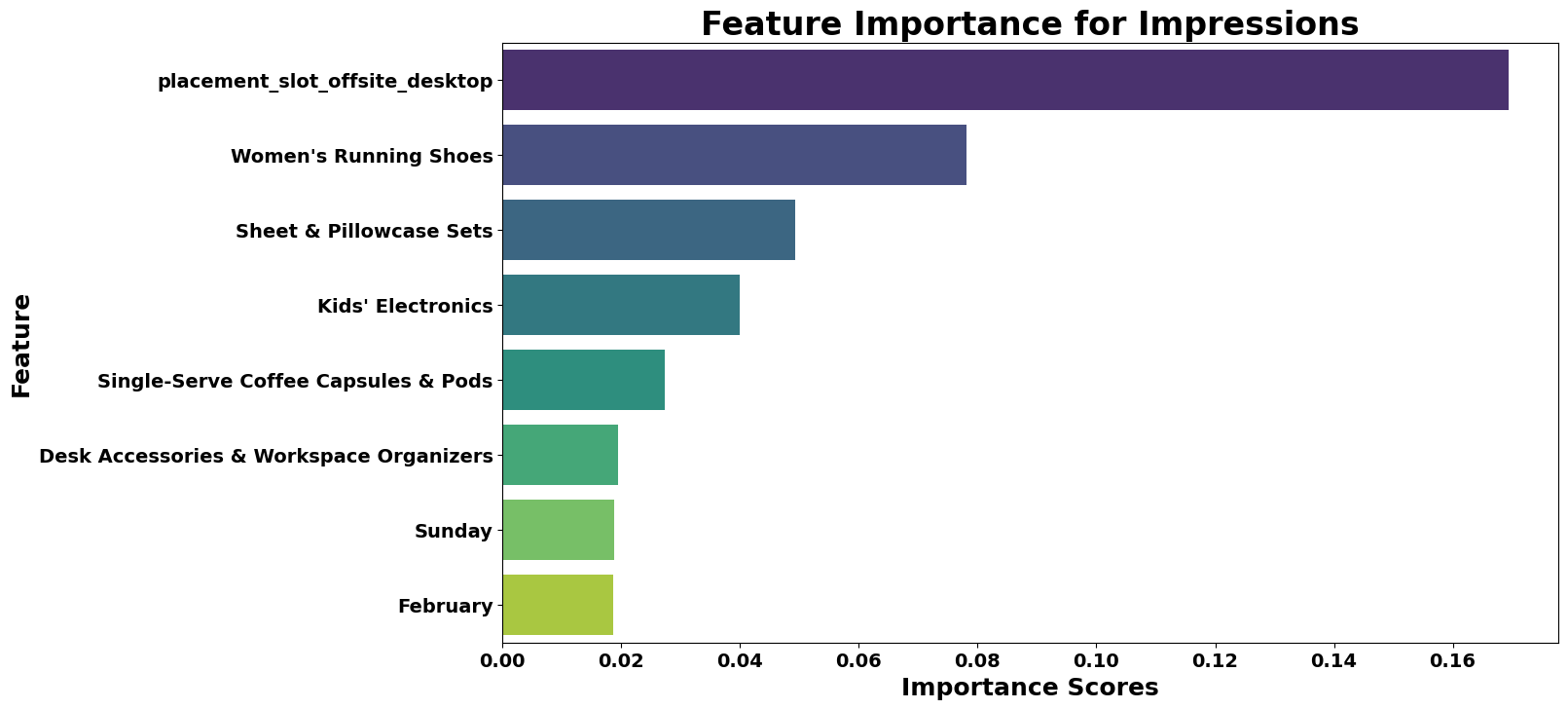
target\_cost = 'adjusted\_cost'



### Approach 2:

For the above approach we are finding features most related to adjusted\_cost but that doesn’t mean it will help to increase reach. Instead if we see the feature importance based on impressions as target bidding can be done based on those features and increasing bidding will lead to increase of reach.

# 



Placement Slot Adjustments:​

Increased bids by 10% for offsite\_desktop placements to capture more high-value traffic.​

​

Top 5 Targeting Categories:​

Applied incremental bid increases based on category importance:​

5% for Women's Running Shoes.​

4% for Sheet & Pillowcase Sets.​

3% for Kids' Electronics.​

2% for Single-Serve Coffee Capsules & Pods.​

1% for Desk Accessories & Workspace Organizers.​

​

Seasonal Adjustments:​

Increased bids by 5% for campaigns running in February to leverage seasonal trends.​

Increased bids by 5% for Sundays to optimize for higher weekend engagement.​

​

* Application of Bidding Strategy: The bidding\_strategy function is applied to each row in the dataset to calculate the optimized\_bid column, which contains the new bid values after applying the adjustments.​
* Calculate Change in Adjusted Cost: A new column adjusted\_cost\_diff is created to capture the difference between the optimized bids and the original adjusted cost.​

**Summary**:

adjusted\_cost optimized\_bid adjusted\_cost\_diff

count 58674.000000 58674.000000 58674.000000

mean 46.754734 51.418755 4.664021

std 40.458269 45.418683 6.375369

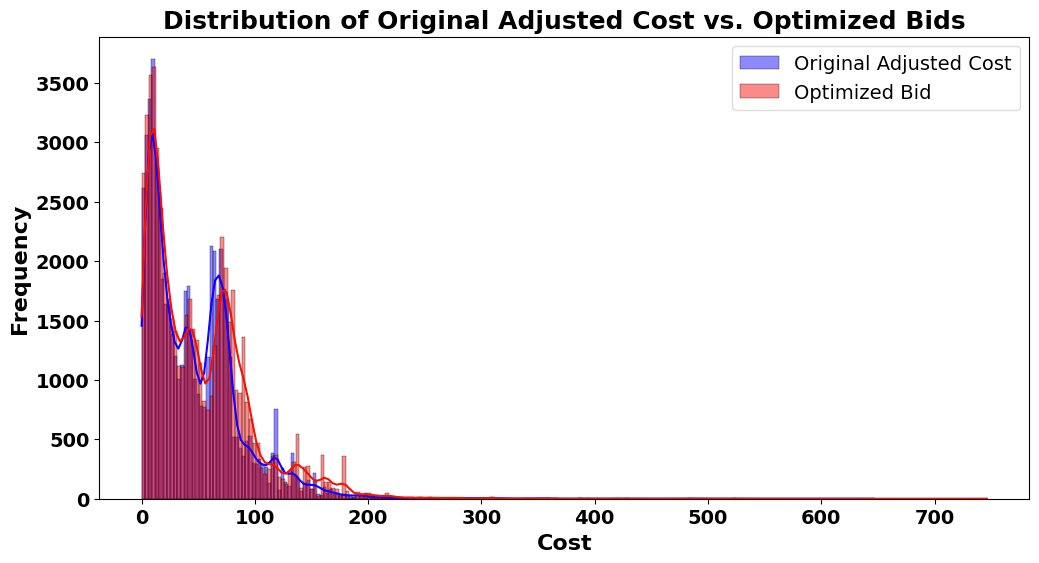
min 0.000140 0.000140 0.000000

25% 13.431010 14.156951 0.233600

50% 39.251250 41.856953 2.128460

75% 69.176000 75.753360 6.622712

max 646.131340 746.281698 100.150358



There is a slight rise of frequency of optimized bid over adjusted cost



Interpretation and Application:

1. **Feature Importance**: The importance\_df will tell us which features have the most impact on impressions.
2. **PDPs**: The PDPs will show how the top features influence impressions, helping us understand their effect in isolation.
3. **SHAP Values**: SHAP summary plots will provide a detailed view of how each feature contributes to the predictions.

Step 2: Generate Partial Dependence Plots

from sklearn.inspection import plot\_partial\_dependence

import matplotlib.pyplot as plt

# Plot PDPs for top features

top\_features = importance\_df.head(5)['Feature'].tolist()

plot\_partial\_dependence(model, X\_train, top\_features, grid\_resolution=50)

plt.show()

Step 3: Calculate SHAP Values

import shap

# Calculate SHAP values

explainer = shap.Explainer(model, X\_train)

shap\_values = explainer(X\_test)

# Plot SHAP summary plot

shap.summary\_plot(shap\_values, X\_test, plot\_type="bar")

# 

# 