Data Preparation Process

- ABC Company Digital Media Data.xslx had missing values and categorical variables that needed attention.
- After visualizing the data, the missing values were replaced with the column mean using R programming language.
- The categorical variables were dummy coded using the Pandas library in Python.
- The continuous variables were also normalized with the use of Python programing language

Modeling: Decision Tree

import library for decision tree

[4]:
```
#import tlib for decision tree
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

Split into features (X) and target variable (y)

[5]:
```
# Split into features (X) and target variable (y)
X = filtered_df.drop('Engaged_Visits_Post_Click', axis=1)
y = filtered_df['Engaged_Visits_Post_Click']
```

split data into test and training set 80:20

[16]:
```
#split data into test and training set 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

import decision tree regressor

[17]:
```
from sklearn.tree import DecisionTreeRegressor
# Create and train the decision tree classifie
# Decision Tree for Classification
regressor = DecisionTreeRegressor()
regressor.fit(X, y)
```

[17]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

[18]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

Make predictions on the test set, calculate MSE on test set and R^2

[19]:
```
y_pred = regressor.predict(X_test)

# Evaluate the model's performance using Mean Squared Error (MSE) and R-squared (R2) score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2) Score: {r2}")
```

```
Mean Squared Error (MSE): 27.11281524885022
R-squared (R2) Score: 0.9940592575037209
```

Find feature importances

[10]:
```
# 1. Feature Importances
feature_importances = regressor.feature_importances_
feature_importances_df = pd.DataFrame({'Feature': X_train.columns, 'Importance': feature_importances})
feature_importances_df = feature_importances_df.sort_values(by='Importance', ascending=False)

# 2. Decision Tree Structure
tree_structure = regressor.tree_

# 3. Predictions on the test set
y_pred = regressor.predict(X_test)

# Evaluate the model's performance using Mean Squared Error (MSE) and R-squared (R2) score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2) Score: {r2}")

print("\nFeature Importances:")
print(feature_importances_df)
print("\nDecision Tree Structure:")
print(tree_structure)
```

```
Mean Squared Error (MSE): 27.11281524885022
R-squared (R2) Score: 0.9940592575037209

Feature Importances:
                  Feature  Importance
1                  Clicks    0.704363
0             Impressions    0.232591
3          Net_Cost:_CM360    0.038369
2   Net_Cost:_Calculated    0.023363
8             Video_Plays    0.000575
7           Video_Replays    0.000396
11               Video_.5    0.000147
13              Video_.25    0.000109
12       Video_Fullscreen    0.000043
9            Video_Pauses    0.000019
6             Video_Skips    0.000018
14      Video_Completions    0.000004
5               Video_.75    0.000002
4           Video_Unmutes    0.000001
10            Video_Mutes    0.000001

Decision Tree Structure:
<sklearn.tree._tree.Tree object at 0x7f572f6bee20>
```

prune tree to prevent overfitting and lower R^2

Find the tree with the best max depth and random state and save that as variables best__tree

[22]:
```
param_grid = {
    'max_depth': np.arange(3, 21),  # Try different depth values from 3 to 20
}

grid_search = GridSearchCV(tree, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

best_tree = grid_search.best_estimator_
print(best_tree)
```

```
DecisionTreeRegressor(max_depth=14, random_state=42)
```

calculate R^2 and MSE of Pruned tree

[24]:
```
y_pred = best_tree.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
pruned_mse = mean_squared_error(y_test, y_pred_pruned)
print("Pruned Mean Squared Error (MSE):", pruned_mse)

# Make predictions on the test set
y_pred = best_tree.predict(X_test)
print("prediction")
print("\n",y_pred)
# Calculate the R-squared score
r2_prune= r2_score(y_test, y_pred)


print("R-squared score pruned:", r2_prune)

#R^2 is high enough for this to be considered a good model, but not too high to expect overfitting.
```

```
Pruned Mean Squared Error (MSE): 533.2064048283789
prediction

 [ 3.62963141  0.59279572  0.59279572 ...  3.62963141  1.37654321
 10.39402318]
R-squared score pruned: 0.8831680915693003
```

Modeling: Gradient Boosted

Gradient Boosted Regression

Import library

```python
[25]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
```

Split the data into training and testing sets

```python
[26]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Create and fit the Gradient Boosted Regressor model

```python
[27]: gb_regressor = GradientBoostingRegressor()
gb_regressor.fit(X_train, y_train)
```

```
[27]: ▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

Make predictions and evaluate the model

```python
[28]: y_pred = gb_regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
```

```python
[29]: print("MSE for Gradient Boosted=. ", mse)
```

```
MSE for Gradient Boosted=.  518.9986803001185
```

```python
[30]: print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 518.9986803001185
```

Grid search with cross-validation

```python
[32]: grid_search = GridSearchCV(gb_regressor, param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

```
[32]: ▸ GridSearchCV
▸ estimator: GradientBoostingRegressor
▸ GradientBoostingRegressor
```

Get the best hyperparameters and model

```python
[33]: best_params = grid_search.best_params_
best_gb_model = grid_search.best_estimator_
```


```python
[32]: grid_search = GridSearchCV(gb_regressor, param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

```
[32]: ▸ GridSearchCV
▸ estimator: GradientBoostingRegressor
▸ GradientBoostingRegressor
```

Get the best hyperparameters and model

```python
[33]: best_params = grid_search.best_params_
best_gb_model = grid_search.best_estimator_
```


Make predictions and calculate the MSE

```python
[34]: y_pred = best_gb_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Best Hyperparameters: {best_params}')
print(f'Mean Squared Error: {mse}')
```

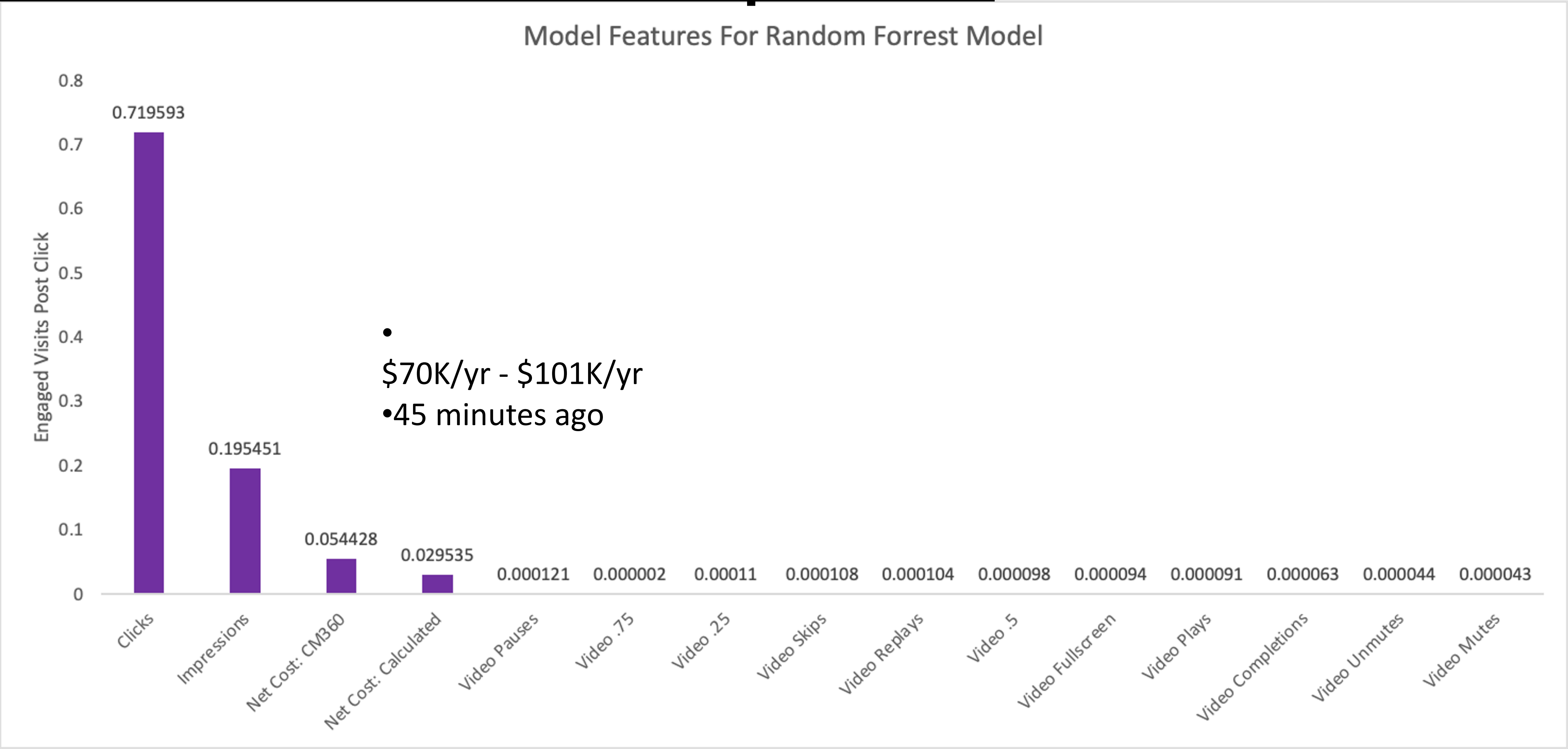
```
Best Hyperparameters: {'max_depth': 6}
Mean Squared Error: 460.0655449488492
```

Modeling: Random Forest

In [9]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

selected_features = df.iloc[:, 1:16]
target = df['Engaged_Visits_Post_Click']

X_train, X_test, y_train, y_test = train_test_split(selected_features, target, test_size=0.2, random_state = 42)
rf_regressor = RandomForestRegressor(n_estimators=100, random_state = 42)
rf_regressor.fit(X_train, y_train)
y_pred = rf_regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r_squared = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r_squared)

feature_importances = rf_regressor.feature_importances_
importance_df = pd.DataFrame({'Feature': selected_features.columns, 'Importance': feature_importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)
print("Feature Importances:")
print(importance_df)
```

```
Mean Squared Error: 389.88302433518857
R-squared: 0.9145719605291046
```

Random Forest Feature Importance



•

$70K/yr - $101K/yr
•45 minutes ago

MINDSHARE