Social Media DataSet Predictions

The dataset is not mine. It is taken from here

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn as sk
import xgboost as xgb
```

```
df = pd.read_csv('data.csv')
df.head()
```

	Account ID	Username	Platform	Follower Count	Posts Per Week	Engagement Rate	Ad Spend (USD)	Conversion Rate	Campaign Reach
0	1	harrislisa	TikTok	54217	3	0.0986	538.10	0.0490	1308
1	2	rhicks	LinkedIn	987518	5	0.0834	479.24	0.0174	13302
2	3	qthomas	Facebook	218870	3	0.1020	150.36	0.0318	11043
3	4	carlosholt	Instagram	207432	6	0.0834	932.62	0.0400	12074

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.shape
```

```
(10000, 9)
```

Main features of the data

- **Account ID:** Unique identifier for each social media account. 🔢
- **Username:** The username of the account. 👤
- **Platform:** Social media platform (Instagram, Twitter, Facebook, TikTok, LinkedIn). 🌐
- **Follower Count:** Number of followers on the account. 📈
- **Posts Per Week:** Average number of posts made per week by the account. 📝
- **Engagement Rate:** The engagement rate calculated as the sum of likes ❤️, comments 💬, and shares 🔄 divided by the follower count.
- **Ad Spend (USD):** Monthly advertising spend in USD for promoting content. 💵
- **Conversion Rate:** Conversion rate, which is the percentage of users who clicked or engaged with the ads. 🔁
- **Campaign Reach:** Number of people reached by the user's campaigns in a given month. 🌍

```
platforms = df['Platform'].unique()
platforms
```

```
array(['TikTok', 'LinkedIn', 'Facebook', 'Instagram', 'Twitter'],
      dtype=object)
```

```
def encode_platform(platform):
  if platform == 'TikTok':
    return 0
  elif platform == 'LinkedIn':
    return 1
  elif platform == 'Instagram':
    return 2
  elif platform == 'Twitter':
    return 3
  elif platform == 'Facebook':
    return 4
  else:
    return None
```

```
df['Platform'] = df['Platform'].apply(encode_platform)
df.head()
```

	Account ID	Username	Platform	Follower Count	Posts Per Week	Engagement Rate	Ad Spend (USD)	Conversion Rate	Campaign Reach
0	1	harrislisa	0	54217	3	0.0986	538.10	0.0490	1308
1	2	rhicks	1	987518	5	0.0834	479.24	0.0174	13302
2	3	qthomas	4	218870	3	0.1020	150.36	0.0318	11043
3	4	carlosholt	2	207432	6	0.0834	932.62	0.0400	12074

Next steps: Generate code with df View recommended plots New interactive sheet

```
df['Ignorance Rate'] = df.apply(lambda row: row['Engagement Rate'] - row['Conversion Rate'], axis=1)
df.head()
```

	Account ID	Username	Platform	Follower Count	Posts Per Week	Engagement Rate	Ad Spend (USD)	Conversion Rate	Campaign Reach	Ignorance Rate
0	1	harrislisa	0	54217	3	0.0986	538.10	0.0490	1308	0.0496
1	2	rhicks	1	987518	5	0.0834	479.24	0.0174	13302	0.0660
2	3	qthomas	4	218870	3	0.1020	150.36	0.0318	11043	0.0702
3	4	carlosholt	2	207432	6	0.0834	932.62	0.0400	12074	0.0434
4	5	parsonsashley	4	350204	2	0.0642	504.44	0.0463	14083	0.0179

Next steps: Generate code with df View recommended plots New interactive sheet

```
label_encoder = sk.preprocessing.LabelEncoder()
df['username_encoded'] = label_encoder.fit_transform(df['Username'])
df.head()
```

	Account ID	Username	Platform	Follower Count	Posts Per Week	Engagement Rate	Ad Spend (USD)	Conversion Rate	Campaign Reach	Ignorance Rate	username_encoded
0	1	harrislisa	0	54217	3	0.0986	538.10	0.0490	1308	0.0496	322
1	2	rhicks	1	987518	5	0.0834	479.24	0.0174	13302	0.0660	733
2	3	qthomas	4	218870	3	0.1020	150.36	0.0318	11043	0.0702	718
3	4	carlosholt	2	207432	6	0.0834	932.62	0.0400	12074	0.0434	121

Next steps: Generate code with df View recommended plots New interactive sheet

Now, we shall split this into training and testing data. We shall be creating models to test for various target variables like campaign reach, follower count, and engagement rate.

Up first, Conversion Rate!

```
x = df.drop(['Conversion Rate', 'Account ID', 'Username'], axis=1)
y = df['Conversion Rate']
x_train, x_test, y_train, y_test = sk.model_selection.train_test_split(x, y, test_size=0.5, random_state=42)
# Linear Regressor
lr = sk.linear_model.LinearRegression()
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
lr_mse = sk.metrics.mean_squared_error(y_test, y_pred)
lr_r2 = sk.metrics.r2_score(y_test, y_pred)
print(f'Linear Regression - Mean Squared Error: {lr_mse}')
print(f'Linear Regression - R2 Score: {lr_r2}')

scores = sk.model_selection.cross_val_score(lr, x, y, cv=5)
print(f'Cross Validation Scores: {scores}')
print(f'Mean Cross Validation Score: {scores.mean()}')
```

```
Linear Regression - Mean Squared Error: 2.548196641216754e-27
Linear Regression - R2 Score: 1.0
Cross Validation Scores: [1. 1. 1. 1. 1.]
Mean Cross Validation Score: 1.0
```

```
# Ridge Regression
from sklearn.linear_model import Ridge
rl = Ridge(alpha=1.0)  # alpha controls the strength of regularization
rl.fit(x_train, y_train)
y_pred = rl.predict(x_test)
rl_mse = sk.metrics.mean_squared_error(y_test, y_pred)
rl_r2 = sk.metrics.r2_score(y_test, y_pred)
print(f'Ridge Regression - Mean Squared Error: {rl_mse}')
print(f'Ridge Regression - R2 Score: {rl_r2}')

scores = sk.model_selection.cross_val_score(rl, x, y, cv=5)
print(f'Cross Validation Scores: {scores}')
print(f'Mean Cross Validation Score: {scores.mean()}')
```

```
Ridge Regression - Mean Squared Error: 6.655730695021878e-05
Ridge Regression - R2 Score: 0.48737430339358934
Cross Validation Scores: [0.59545427 0.60936648 0.58961271 0.60295254 0.60278472]
Mean Cross Validation Score: 0.6000341433713199
```

```
# Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(random_state=42)
dtr.fit(x_train, y_train)
y_pred = dtr.predict(x_test)
dtr_mse = sk.metrics.mean_squared_error(y_test, y_pred)
dtr_r2 = sk.metrics.r2_score(y_test, y_pred)
print(f'Decision Tree Regressor - Mean Squared Error: {dtr_mse}')
print(f'Decision Tree Regressor - R2 Score: {dtr_r2}')

scores = sk.model_selection.cross_val_score(dtr, x, y, cv=5)
print(f'Cross Validation Scores: {scores}')
print(f'Mean Cross Validation Score: {scores.mean()}')
```

```
Decision Tree Regressor - Mean Squared Error: 4.4299999999999998e-09
Decision Tree Regressor - R2 Score: 0.9999658800522433
Cross Validation Scores: [0.99998992 0.9999906  0.99998714 0.9999908  0.99998892]
Mean Cross Validation Score: 0.9999894758223047
```

```
# Gradient Boosted Regressor
gbr = xgb.XGBRegressor(n_estimators=100, random_state=42)
gbr.fit(x_train, y_train)
y_pred = gbr.predict(x_test)
gbr_mse = sk.metrics.mean_squared_error(y_test, y_pred)
gbr_r2 = sk.metrics.r2_score(y_test, y_pred)
print(f'Gradient Boosted Regressor - Mean Squared Error: {gbr_mse}')
print(f'Gradient Boosted Regressor - R2 Score: {gbr_r2}')

scores = sk.model_selection.cross_val_score(dtr, x, y, cv=5)
print(f'Cross Validation Scores: {scores}')
print(f'Mean Cross Validation Score: {scores.mean()}')
```

```
Gradient Boosted Regressor - Mean Squared Error: 7.69425840092344e-08
Gradient Boosted Regressor - R2 Score: 0.9994073866937565
Cross Validation Scores: [0.99998992 0.9999906  0.99998714 0.9999908  0.99998892]
Mean Cross Validation Score: 0.9999894758223047
```

```
# Support Vector Regressor
from sklearn.svm import SVR
svr = SVR(kernel='rbf', C=100, epsilon=0.1)
scaler = sk.preprocessing.StandardScaler()
svr.fit(scaler.fit_transform(x_train), y_train)
y_pred = svr.predict(scaler.transform(x_test))
svr_mse = sk.metrics.mean_squared_error(y_test, y_pred)
svr_r2 = sk.metrics.r2_score(y_test, y_pred)
print(f'Support Vector Regressor - Mean Squared Error: {svr_mse}')
print(f'Support Vector Regressor - R2 Score: {svr_r2}')

scores = sk.model_selection.cross_val_score(svr, x, y, cv=5)
print(f'Cross Validation Scores: {scores}')
print(f'Mean Cross Validation Score: {scores.mean()}')
```

```
Support Vector Regressor - Mean Squared Error: 0.00013056788400000002
Support Vector Regressor - R2 Score: -0.005636429069866722
Cross Validation Scores: [-0.00419068 -0.00703223 -0.00364542 -0.00901725 -0.00303763]
Mean Cross Validation Score: -0.005384639690162185
```

```
# Multi Layer Perceptron Regressor
from sklearn.neural_network import MLPRegressor
mlpr = MLPRegressor(hidden_layer_sizes=(500, 500), max_iter=1000)
mlpr.fit(x_train, y_train)
y_pred = mlpr.predict(x_test)
```

```
mlpr_mse = sk.metrics.mean_squared_error(y_test, y_pred)
mlpr_r2 = sk.metrics.r2_score(y_test, y_pred)
print(f'Multi Layer Perceptron Regressor - Mean Squared Error: {mlpr_mse}')
print(f'Multi Layer Perceptron Regressor - R2 Score: {mlpr_r2}')

scores = sk.model_selection.cross_val_score(svr, x, y, cv=5)
print(f'Cross Validation Scores: {scores}')
print(f'Mean Cross Validation Score: {scores.mean()}')
```

```
Multi Layer Perceptron Regressor - Mean Squared Error: 255.2314312898431
Multi Layer Perceptron Regressor - R2 Score: -1965796.5398353613
Cross Validation Scores: [-0.00419068 -0.00703223 -0.00364542 -0.00901725 -0.00303763]
Mean Cross Validation Score: -0.005384639690162185
```

So, the Linear Regressor overfitted the data exhibiting R2 Score of 1.0 so excluding that, The Decision Tree Regressor and the Gradient Booster Regressor exhibited 95+ % Mean Cross Validation Scores which makes them the top models to use for this dataset.

Start coding or generate with AI.