

```

import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import numpy as np
from scipy import stats
from scipy.stats import shapiro, f_oneway, skew, kurtosis, wilcoxon,
ttest_rel, chi2_contingency
import statsmodels.api as sm
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('./data.csv')
df.head(10)

```

	Timestamp \
0	2025/06/21 5:22:28 pm EEST
1	2025/06/21 5:29:05 pm EEST
2	2025/06/21 5:59:31 pm EEST
3	2025/06/21 6:36:08 pm EEST
4	2025/06/21 7:41:10 pm EEST
5	2025/06/21 7:41:13 pm EEST
6	2025/06/21 7:43:09 pm EEST
7	2025/06/21 9:25:23 pm EEST
8	2025/06/21 10:18:19 pm EEST
9	2025/06/21 10:32:55 pm EEST

Are you 18 years or older and willing to participate voluntarily?

\	
0	Yes
1	Yes
2	Yes
3	Yes
4	Yes
5	Yes
6	Yes
7	Yes
8	Yes
9	Yes

	What is your age group?	What is your gender?	\
0	23-27	Male	
1	23-27	Male	
2	28-50	Female	
3	23-27	Male	
4	23-27	Female	
5	23-27	Female	
6	23-27	Female	
7	28-50	Female	
8	23-27	Male	
9	28-50	Female	

	What is your highest level of education?	\
0	Master's	
1	Master's	
2	Master's	
3	Master's	
4	Master's	
5	Bachelor's	
6	Master's	
7	Master's	
8	Bachelor's	
9	Bachelor's	

	How often do you watch video ads on platforms like YouTube, Instagram, or TikTok?	\
0		5
1		5
2		5
3		4
4		1
5		3
6		3
7		5
8		3
9		2

	How familiar are you with AI-generated content (e.g., AI-made videos/images)?	\
0		4

1	5
2	5
3	5
4	5
5	5
6	5
7	4
8	5
9	1

	Ad: A	Ad: B	The ad feels original and creative?_A	...	\
0	3	5	3	...	
1	3	5	3	...	
2	3	3	3	...	
3	3	4	3	...	
4	3	5	5	...	
5	3	3	3	...	
6	2	4	2	...	
7	3	3	3	...	
8	4	5	3	...	
9	3	2	1	...	

	The ad presents the product in an imaginative way?_B	\
0	5	
1	4	
2	4	
3	4	
4	5	
5	3	
6	3	
7	3	
8	2	
9	1	

	The ad feels genuine and trustworthy?_B	\
0	5	
1	5	
2	4	
3	3	
4	5	
5	3	

6	3
7	3
8	2
9	1

The ad aligns well with what I expect from a fashion brand?\_B \

0	5
1	4
2	4
3	5
4	5
5	3
6	4
7	3
8	2
9	1

This ad appears to have been produced quickly?\_B \

0	5
1	2
2	4
3	3
4	1
5	4
6	3
7	4
8	5
9	4

The visual quality suggests minimal production time?\_B \

0	5
1	2
2	4
3	4
4	1
5	3
6	4
7	4
8	5
9	4

This ad seems inexpensive to produce?\_B \

0	5
1	1
2	4
3	3
4	1
5	4
6	3
7	4

8	5
9	4

I believe less effort or budget went into making this ad?_B \	
0	5
1	2
2	4
3	2
4	1
5	3
6	2
7	4
8	5
9	4

Which ad did you prefer overall? \	
0	Ad B
1	Ad B
2	Ad A
3	Ad B
4	Ad A
5	Ad B
6	Ad B
7	Ad B
8	Ad B
9	Ad A

Which ad do you think was created using AI? \	
0	Ad B
1	Ad A
2	Ad B
3	Ad A
4	Ad A
5	Ad A
6	Ad A
7	Ad A
8	Ad B
9	Ad A

Would you mind if ads were created using AI as long as they are effective?	
0	No
1	No
2	Yes
3	No
4	Yes

5	No
6	No
7	No
8	Yes
9	No

[10 rows x 28 columns]

df.columns

```
Index(['Timestamp',
      'Are you 18 years or older and willing to participate
voluntarily? ',
      'What is your age group? ', 'What is your gender? ',
      'What is your highest level of education? ',
      'How often do you watch video ads on platforms like YouTube,
Instagram, or TikTok? ',
      'How familiar are you with AI-generated content (e.g., AI-made
videos/images)? ',
      'Ad: A ', 'Ad: B ', 'The ad feels original and creative?_A',
      'The ad presents the product in an imaginative way?_A',
      'The ad feels genuine and trustworthy?_A',
      'The ad aligns well with what I expect from a fashion brand?
_A',
      'This ad appears to have been produced quickly?_A',
      'The visual quality suggests minimal production time?_A',
      'This ad seems inexpensive to produce?_A',
      'I believe less effort or budget went into making this ad?_A',
      'The ad feels original and creative?_B',
      'The ad presents the product in an imaginative way?_B',
      'The ad feels genuine and trustworthy?_B',
      'The ad aligns well with what I expect from a fashion brand?
_B',
      'This ad appears to have been produced quickly?_B',
      'The visual quality suggests minimal production time?_B',
      'This ad seems inexpensive to produce?_B',
      'I believe less effort or budget went into making this ad?_B',
      'Which ad did you prefer overall? ',
      'Which ad do you think was created using AI? ',
      'Would you mind if ads were created using AI as long as they
are effective? '],
      dtype='object')
```

```
df = df[['What is your age group? ', 'What is your gender? ',
        'What is your highest level of education? '],
```

```

    'How often do you watch video ads on platforms like YouTube,
    Instagram, or TikTok? ',
    'How familiar are you with AI-generated content (e.g., AI-made
    videos/images)? ', 'The ad feels original and creative?_A',
    'The ad presents the product in an imaginative way?_A',
    'The ad feels genuine and trustworthy?_A',
    'The ad aligns well with what I expect from a fashion brand?
    _A',
    'This ad appears to have been produced quickly?_A',
    'The visual quality suggests minimal production time?_A',
    'This ad seems inexpensive to produce?_A',
    'I believe less effort or budget went into making this ad?_A',
    'The ad feels original and creative?_B',
    'The ad presents the product in an imaginative way?_B',
    'The ad feels genuine and trustworthy?_B',
    'The ad aligns well with what I expect from a fashion brand?
    _B',
    'This ad appears to have been produced quickly?_B',
    'The visual quality suggests minimal production time?_B',
    'This ad seems inexpensive to produce?_B',
    'I believe less effort or budget went into making this ad?_B',
    'Which ad did you prefer overall? ',
    'Which ad do you think was created using AI? ']]

```

```
len(df.columns)
```

23

```

dff = pd.DataFrame({
    'age': df[' What is your age group? '], 'gender': df['What is
    your gender? '], 'ad_exposure': df['How often do you watch video ads
    on platforms like YouTube, Instagram, or TikTok? '], 'ai_exposure':
    df['How familiar are you with AI-generated content (e.g., AI-made
    videos/images)? '], 'creativity_a': df[['The ad feels original and
    creative?_A', 'The ad presents the product in an imaginative way?
    _A']].mean(axis=1), 'creativity_b': df[['The ad feels original and
    creative?_B', 'The ad presents the product in an imaginative way?
    _B']].mean(axis=1), 'authenticity_a': df[['The ad feels genuine and
    trustworthy?_A', 'The ad aligns well with what I expect from a
    fashion brand?_A']].mean(axis=1), 'authenticity_b': df[['The ad feels
    genuine and trustworthy?_B', 'The ad aligns well with what I expect
    from a fashion brand?_B']].mean(axis=1), 'production_speed_a':
    df[['This ad appears to have been produced quickly?_A', 'The visual
    quality suggests minimal production time?_A']].mean(axis=1),
    'production_speed_b': df[['This ad appears to have been produced
    quickly?_B', 'The visual quality suggests minimal production time?
    _B']].mean(axis=1), 'cost_eff_a': df[['This ad seems inexpensive to
    produce?_A', 'I believe less effort or budget went into making this
    ad?_A']].mean(axis=1), 'cost_eff_b': df[['This ad seems inexpensive
    to produce?_B', 'I believe less effort or budget went into making this

```

```
ad?_B']]).mean(axis=1), 'ad_preference' : df['Which ad did you prefer
overall? '], 'ad_prediction': df['Which ad do you think was created
using AI? '])
})
```

```
dff.columns
```

```
Index(['age', 'gender', 'ad_exposure', 'ai_exposure', 'creativity_a',
      'creativity_b', 'authenticity_a', 'authenticity_b',
      'production_speed_a', 'production_speed_b', 'cost_eff_a',
      'cost_eff_b',
      'ad_preference', 'ad_prediction'],
      dtype='object')
```

```
dff.head()
```

	age	gender	ad_exposure	ai_exposure	creativity_a	creativity_b
0	23-27	Male	5	4	4.0	5.0
1	23-27	Male	5	5	3.0	4.5
2	28-50	Female	5	5	3.0	4.0
3	23-27	Male	4	5	3.0	4.0
4	23-27	Female	1	5	5.0	5.0

	authenticity_a	authenticity_b	production_speed_a
0	4.0	5.0	5.0
1	3.0	4.5	4.5
2	4.0	4.0	4.0
3	2.5	4.0	4.5
4	5.0	5.0	1.0

	cost_eff_a	cost_eff_b	ad_preference	ad_prediction
0	3.0	5.0	Ad B	Ad B
1	5.0	1.5	Ad B	Ad A
2	3.5	4.0	Ad A	Ad B
3	4.5	2.5	Ad B	Ad A
4	1.0	1.0	Ad A	Ad A

```
dff['age'].unique()
```



```

array(['23-27', '28-50', '18-22', '51 and above'], dtype=object)
dff['gender'].unique()
array(['Male', 'Female', 'Other'], dtype=object)
ad_mapping = {'Ad A': 1, 'Ad B': 2}

dff['ad_preference'] = dff['ad_preference'].map(ad_mapping)
dff['ad_prediction'] = dff['ad_prediction'].map(ad_mapping)

age_mapping = {'18-22': 1, '23-27': 2, '28-50': 3, '51 and above': 4}
gender_mapping = {'Male': 1, 'Female': 2, 'Other': 3}

dff['age'] = dff['age'].map(age_mapping)
dff['gender'] = dff['gender'].map(gender_mapping)

dff.head()

```

	age	gender	ad_exposure	ai_exposure	creativity_a	creativity_b
0	2	1	5	4	4.0	5.0
1	2	1	5	5	3.0	4.5
2	3	2	5	5	3.0	4.0
3	2	1	4	5	3.0	4.0
4	2	2	1	5	5.0	5.0

	authenticity_a	authenticity_b	production_speed_a	production_speed_b
0	4.0	5.0	5.0	5.0
1	3.0	4.5	4.5	2.0
2	4.0	4.0	4.0	4.0
3	2.5	4.0	4.5	3.5
4	5.0	5.0	1.0	1.0

	cost_eff_a	cost_eff_b	ad_preference	ad_prediction
0	3.0	5.0	2	2
1	5.0	1.5	2	1
2	3.5	4.0	1	2
3	4.5	2.5	2	1
4	1.0	1.0	1	1

## data mean

```
dff.mean()

age                2.305556
gender             1.488889
ad_exposure        4.122222
ai_exposure        4.194444
creativity_a       3.669444
creativity_b       3.730556
authenticity_a     3.541667
authenticity_b     3.647222
production_speed_a 3.333333
production_speed_b 3.172222
cost_eff_a         3.247222
cost_eff_b         2.980556
ad_preference      1.500000
ad_prediction      1.494444
dtype: float64
```

## data standard deviation

```
dff.std()

age                0.832961
gender             0.533659
ad_exposure        0.961002
ai_exposure        1.008884
creativity_a       1.141111
creativity_b       1.093868
authenticity_a     1.040889
authenticity_b     1.047339
production_speed_a 1.160331
production_speed_b 1.087412
cost_eff_a         1.213860
cost_eff_b         1.221735
ad_preference      0.501395
ad_prediction      0.501364
dtype: float64

dff[['ad_exposure', 'ai_exposure', 'creativity_a',
      'creativity_b', 'authenticity_a', 'authenticity_b',
      'production_speed_a', 'production_speed_b', 'cost_eff_a',
      'cost_eff_b']].corr()

           ad_exposure  ai_exposure  creativity_a
creativity_b \
ad_exposure          1.000000      0.274981      0.421676
```

0.167021			
ai_exposure	0.274981	1.000000	0.211427
0.265415			
creativity_a	0.421676	0.211427	1.000000
0.071467			
creativity_b	0.167021	0.265415	0.071467
1.000000			
authenticity_a	0.410958	0.242276	0.778320
0.112953			
authenticity_b	0.159640	0.282053	0.026924
0.834539			
production_speed_a	0.206246	0.213956	-0.116733
0.180094			
production_speed_b	0.198930	0.027866	0.130551
0.203821			
cost_eff_a	0.136782	0.179495	-0.317778
0.263973			
cost_eff_b	0.159057	0.000818	0.231789
0.242218			

	authenticity_a	authenticity_b	production_speed_a
\			
ad_exposure	0.410958	0.159640	0.206246
ai_exposure	0.242276	0.282053	0.213956
creativity_a	0.778320	0.026924	-0.116733
creativity_b	0.112953	0.834539	0.180094
authenticity_a	1.000000	0.072491	-0.132984
authenticity_b	0.072491	1.000000	0.242110
production_speed_a	-0.132984	0.242110	1.000000
production_speed_b	0.084935	-0.167094	0.298496
cost_eff_a	-0.325444	0.320559	0.771134
cost_eff_b	0.133530	-0.248795	0.162231

	production_speed_b	cost_eff_a	cost_eff_b
ad_exposure	0.198930	0.136782	0.159057
ai_exposure	0.027866	0.179495	0.000818
creativity_a	0.130551	-0.317778	0.231789
creativity_b	-0.203821	0.263973	-0.242218
authenticity_a	0.084935	-0.325444	0.133530
authenticity_b	-0.167094	0.320559	-0.248795

production_speed_a	0.298496	0.771134	0.162231
production_speed_b	1.000000	0.261714	0.727914
cost_eff_a	0.261714	1.000000	0.134165
cost_eff_b	0.727914	0.134165	1.000000

```

correlation_matrix = dff[['ad_exposure', 'ai_exposure',
'creativity_a',
'creativity_b', 'authenticity_a',
'authenticity_b',
'production_speed_a', 'production_speed_b',
'cost_eff_a',
'cost_eff_b']].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix,
            annot=True,          # Show correlation values
            cmap='coolwarm',     # Color scheme (blue-white-red)
            center=0,            # Center colormap at 0
            square=True,         # Make cells square
            fmt='.2f',            # Format numbers to 2 decimal places
            (better for larger matrix)
            cbar_kws={'shrink': 0.8}, # Adjust colorbar size
            linewidths=0.5,        # Add lines between cells
            annot_kws={'size': 9}) # Smaller text size for
readability

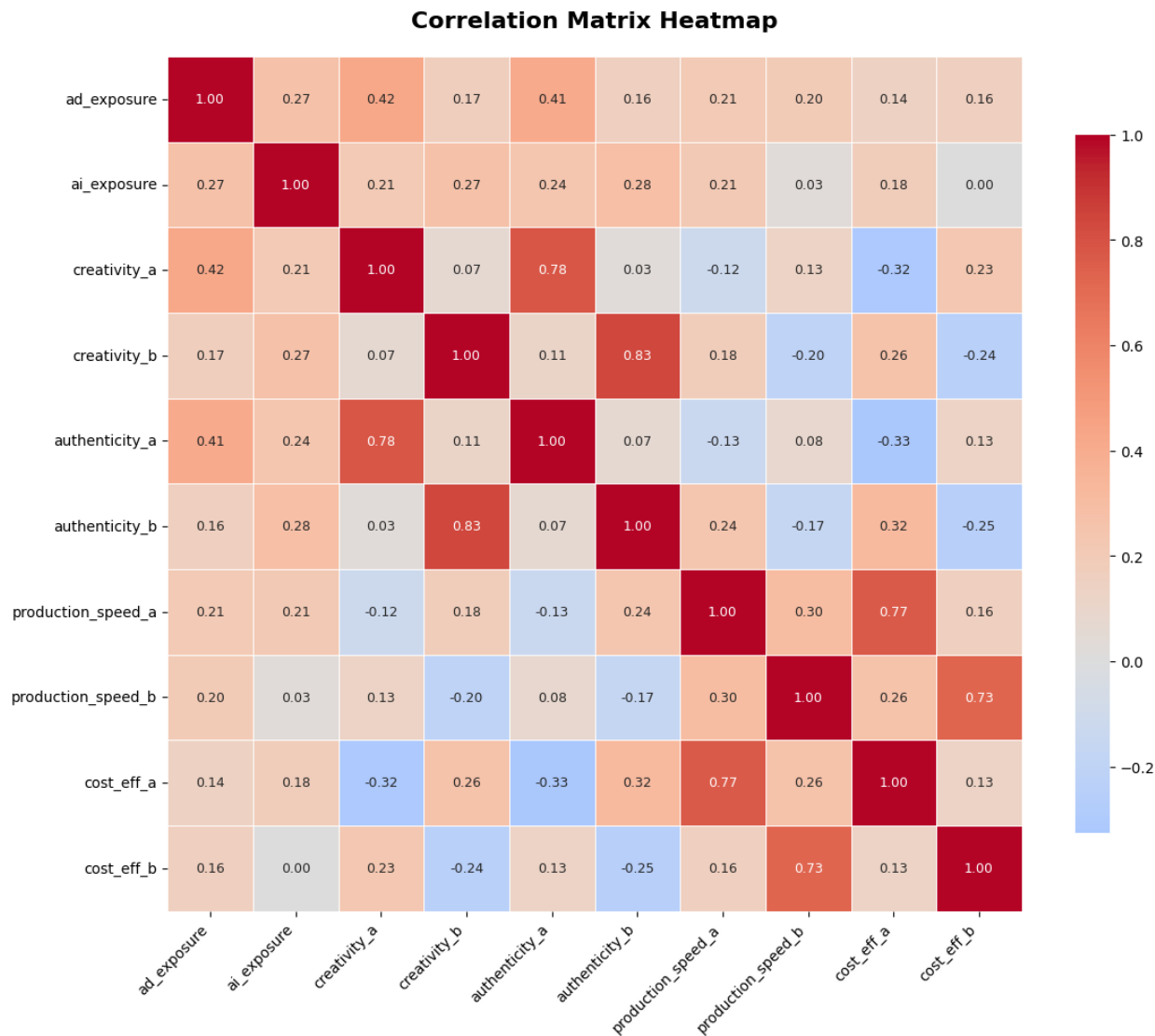
plt.title('Correlation Matrix Heatmap', fontsize=16,
fontweight='bold', pad=20)
plt.xlabel('') # Remove x-axis label (redundant with tick labels)
plt.ylabel('') # Remove y-axis label (redundant with tick labels)
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better
readability
plt.yticks(rotation=0)              # Keep y-axis labels horizontal

plt.tight_layout()

plt.show()

print("Correlation Matrix:")
print(correlation_matrix.round(3))

```



Correlation Matrix:

	ad_exposure	ai_exposure	creativity_a
creativity_b \			
ad_exposure	1.000	0.275	0.422
ai_exposure	0.275	1.000	0.211
creativity_a	0.422	0.211	1.000
creativity_b	0.167	0.265	0.071
authenticity_a	0.411	0.242	0.778
authenticity_b	0.160	0.282	0.027
production_speed_a	0.206	0.214	-0.117

0.180				
production_speed_b	0.199	0.028	0.131	-
0.204				
cost_eff_a	0.137	0.179	-0.318	
0.264				
cost_eff_b	0.159	0.001	0.232	-
0.242				

	authenticity_a	authenticity_b	production_speed_a
\			
ad_exposure	0.411	0.160	0.206
ai_exposure	0.242	0.282	0.214
creativity_a	0.778	0.027	-0.117
creativity_b	0.113	0.835	0.180
authenticity_a	1.000	0.072	-0.133
authenticity_b	0.072	1.000	0.242
production_speed_a	-0.133	0.242	1.000
production_speed_b	0.085	-0.167	0.298
cost_eff_a	-0.325	0.321	0.771
cost_eff_b	0.134	-0.249	0.162

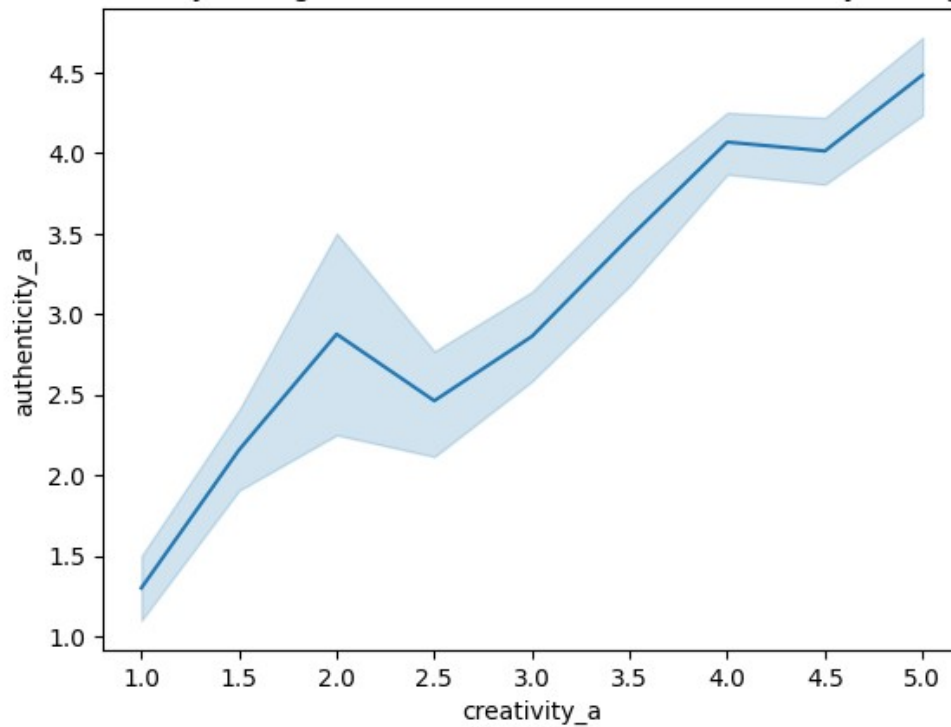
	production_speed_b	cost_eff_a	cost_eff_b
ad_exposure	0.199	0.137	0.159
ai_exposure	0.028	0.179	0.001
creativity_a	0.131	-0.318	0.232
creativity_b	-0.204	0.264	-0.242
authenticity_a	0.085	-0.325	0.134
authenticity_b	-0.167	0.321	-0.249
production_speed_a	0.298	0.771	0.162
production_speed_b	1.000	0.262	0.728
cost_eff_a	0.262	1.000	0.134
cost_eff_b	0.728	0.134	1.000

```

sns.lineplot(data=dff, x='creativity_a', y='authenticity_a')
plt.title('Perceived creativity of AI-generated Ad vs Perceived authenticity of AI-generated Ad')
plt.show()

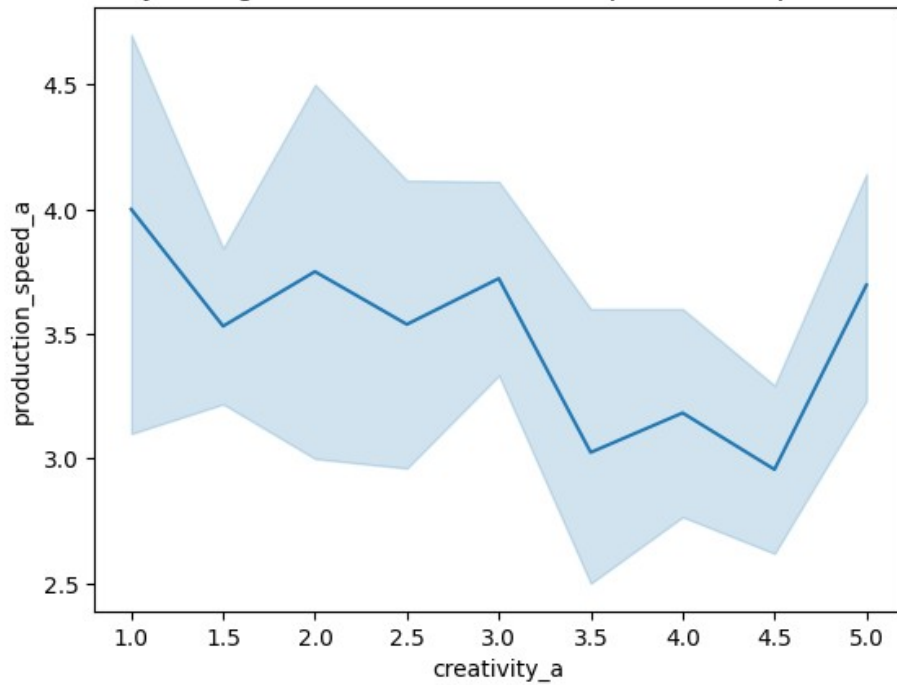
```

Perceived creativity of AI-generated Ad vs Perceived authenticity of AI-generated Ad



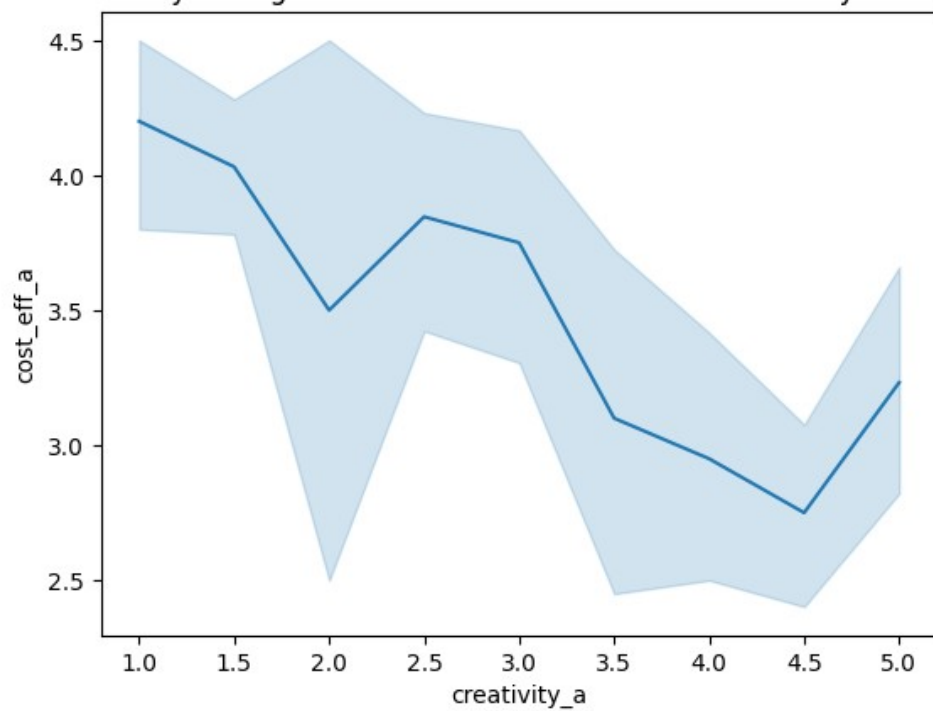
```
sns.lineplot(data=dff, x='creativity_a', y='production_speed_a')  
plt.title('Perceived creativity of AI-generated Ad vs Perceived  
production speed of AI-generated Ad')  
plt.show()
```

Perceived creativity of AI-generated Ad vs Perceived production speed of AI-generated Ad



```
sns.lineplot(data=dff, x='creativity_a', y='cost_eff_a')  
plt.title('Perceived creativity of AI-generated Ad vs Perceived cost  
efficiency of AI-generated Ad')  
plt.show()
```

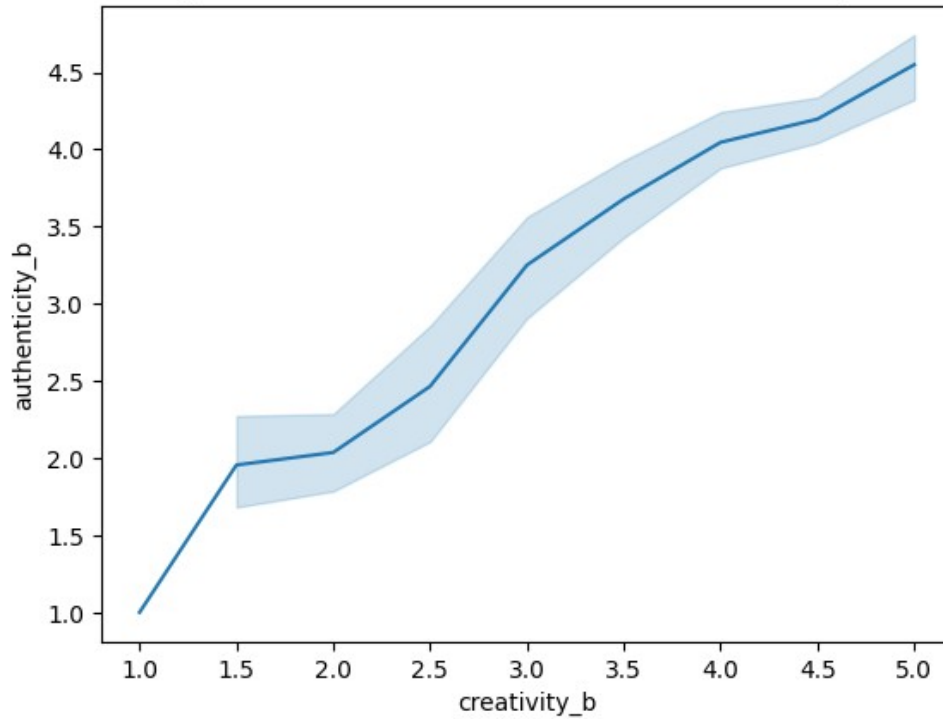
Perceived creativity of AI-generated Ad vs Perceived cost efficiency of AI-generated Ad





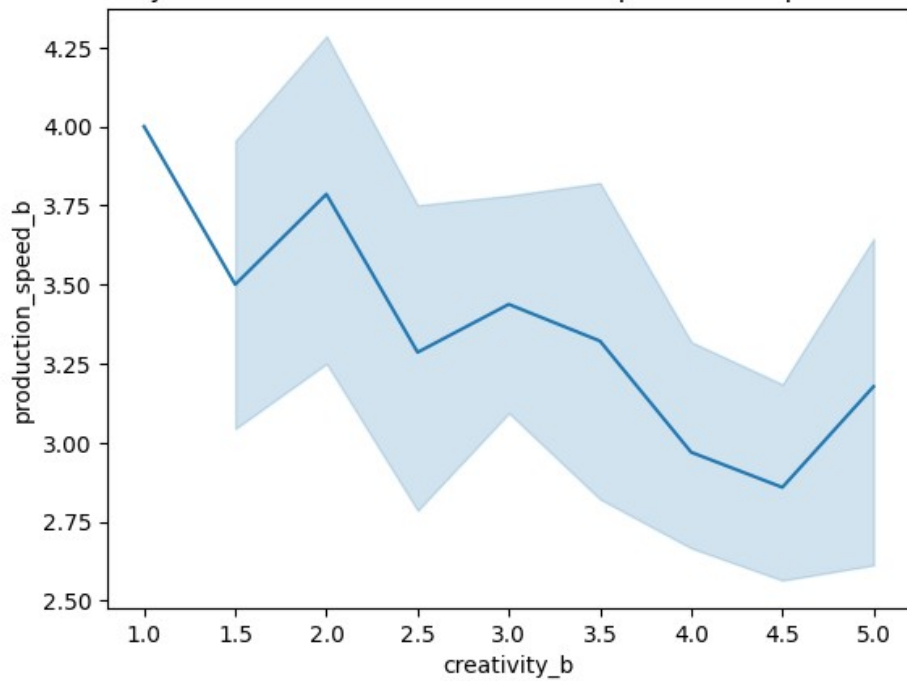
```
sns.lineplot(data=dff, x='creativity_b', y='authenticity_b')
plt.title('Perceived creativity of human-made Ad vs Perceived
authenticity of human-made Ad')
plt.show()
```

Perceived creativity of human-made Ad vs Perceived authenticity of human-made Ad



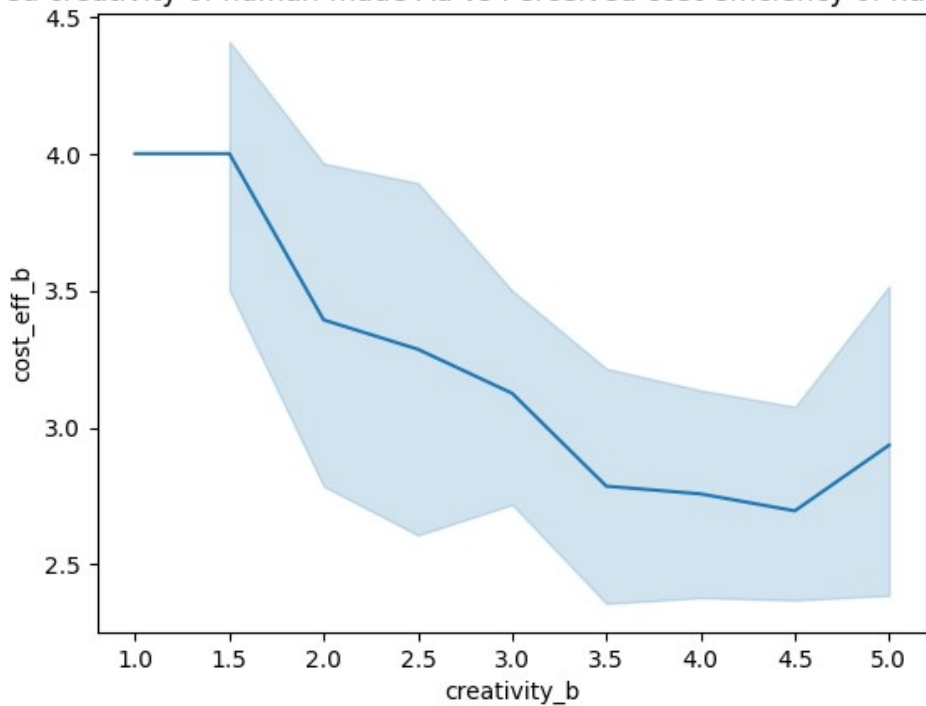
```
sns.lineplot(data=dff, x='creativity_b', y='production_speed_b')
plt.title('Perceived creativity of human-made Ad vs Perceived
production speed of human-made Ad')
plt.show()
```

Perceived creativity of human-made Ad vs Perceived production speed of human-made Ad



```
sns.lineplot(data=dff, x='creativity_b', y='cost_eff_b')  
plt.title('Perceived creativity of human-made Ad vs Perceived cost  
efficiency of human-made Ad')  
plt.show()
```

Perceived creativity of human-made Ad vs Perceived cost efficiency of human-made Ad



## Overall advertisement preference and prediction breakdown

```
dff['ad_preference'].value_counts()
```

```
ad_preference
```

```
2    90
```

```
1    90
```

```
Name: count, dtype: int64
```

```
dff['ad_prediction'].value_counts()
```

```
ad_prediction
```

```
1    91
```

```
2    89
```

```
Name: count, dtype: int64
```

## Advertisement prediction breakdown by age and ai exposure

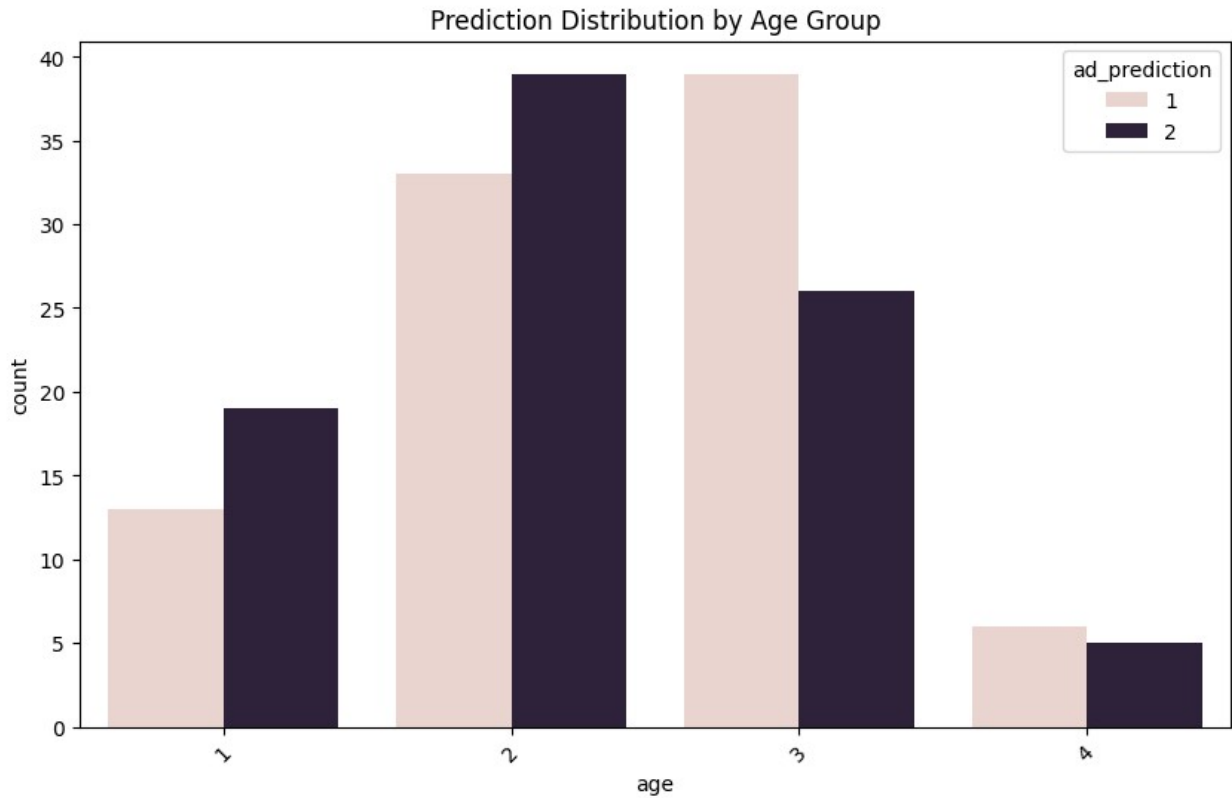
```
dff.groupby(['age', 'ai_exposure'])['ad_prediction'].value_counts()
```

age	ai_exposure	ad_prediction	
1	2	1	2
		2	1
	3	2	4
		1	2
	4	1	5
2	4	2	2
		2	12
	5	2	4
		1	2
	2	2	2
		3	8
	3	1	5
		2	10
	5	1	8
		2	20
3	1	2	19
		1	2
	2	2	1
		1	1
	3	1	4
		2	1
	4	1	14
		2	7
	5	1	18
		2	1

		2	17
4	1	2	1
	2	1	2
		2	1
	3	1	1
		2	1
	4	1	3
		2	1
	5	2	1

Name: count, dtype: int64

```
plt.figure(figsize=(10, 6))
sns.countplot(data=dff, x='age', hue='ad_prediction')
plt.title('Prediction Distribution by Age Group')
plt.xticks(rotation=45)
plt.show()
```



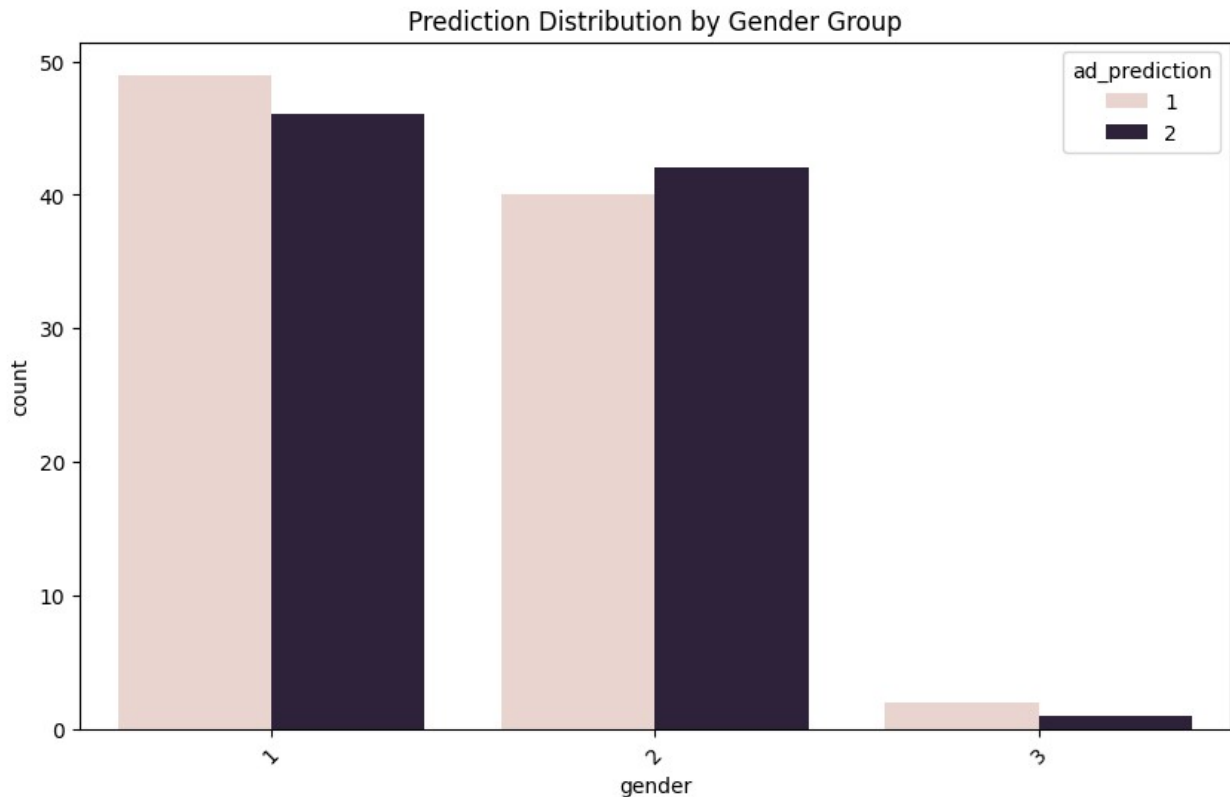
## Advertisement prediction breakdown by gender and ai exposure

```
dff.groupby(['gender', 'ai_exposure'])['ad_prediction'].value_counts()
```

gender	ai_exposure	ad_prediction	
1	2	1	2
		2	2
	3	1	8
		2	5
	4	1	13
		2	10
	5	2	29
		1	26
2	1	1	2
		2	2
	2	1	3
		2	2
	3	2	8
		1	3
	4	1	17
		2	10
	5	2	20
		1	15
3	3	1	1
		2	1
	5	1	1

Name: count, dtype: int64

```
plt.figure(figsize=(10, 6))
sns.countplot(data=dff, x='gender', hue='ad_prediction')
plt.title('Prediction Distribution by Gender Group')
plt.xticks(rotation=45)
plt.show()
```



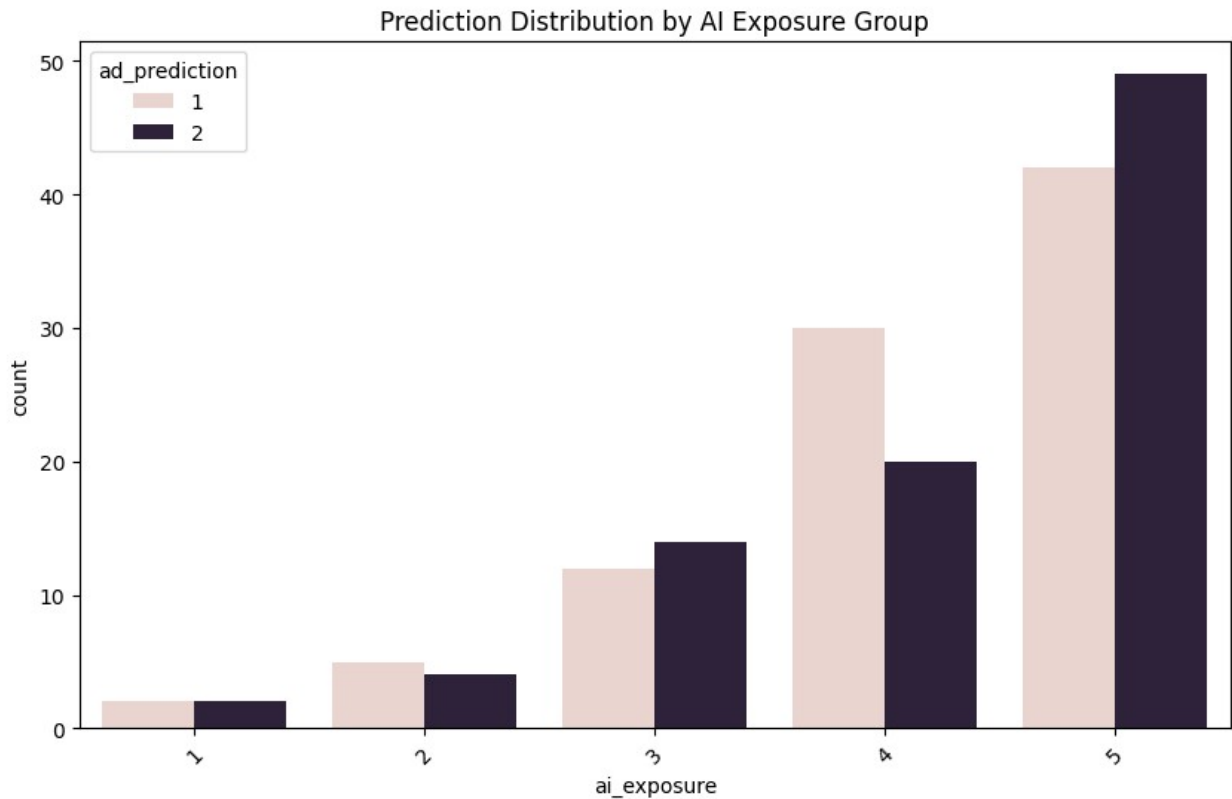
## Advertisement prediction breakdown by ai exposure

```
dff.groupby('ai_exposure')['ad_prediction'].value_counts()
```

ai_exposure	ad_prediction	count
1	1	2
	2	2
2	1	5
	2	4
3	2	14
	1	12
4	1	30
	2	20
5	2	49
	1	42

Name: count, dtype: int64

```
plt.figure(figsize=(10, 6))
sns.countplot(data=dff, x='ai_exposure', hue='ad_prediction')
plt.title('Prediction Distribution by AI Exposure Group')
plt.xticks(rotation=45)
plt.show()
```



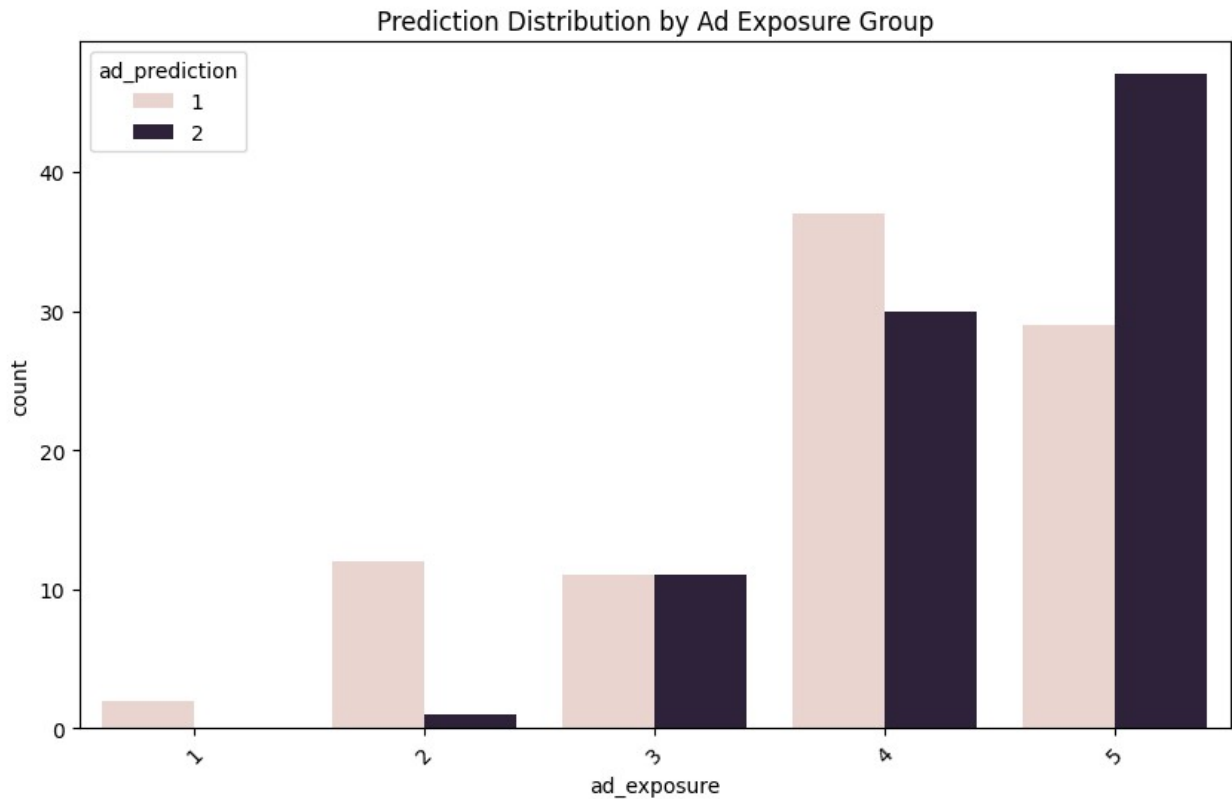
## Advertisement prediction breakdown by ad exposure

```
dff.groupby('ad_exposure')['ad_prediction'].value_counts()
```

```
ad_exposure  ad_prediction
1            1              2
2            1             12
            2              1
3            1             11
            2             11
4            1             37
            2             30
5            2             47
            1             29
```

Name: count, dtype: int64

```
plt.figure(figsize=(10, 6))
sns.countplot(data=dff, x='ad_exposure', hue='ad_prediction')
plt.title('Prediction Distribution by Ad Exposure Group')
plt.xticks(rotation=45)
plt.show()
```



## Quantitative Analysis

```
feature_pairs = [
    ('creativity_a', 'creativity_b'),
    ('authenticity_a', 'authenticity_b'),
    ('production_speed_a', 'production_speed_b'),
    ('cost_eff_a', 'cost_eff_b')
]

# ===== CRONBACH'S ALPHA =====
def cronbachs_alpha(data):
    """
    Calculate Cronbach's Alpha for internal consistency reliability

    Parameters:
    data: DataFrame or array-like with items as columns

    Returns:
    float: Cronbach's alpha coefficient
    """
    # Number of items
    k = data.shape[1]

    # Variance of each item
```



```

item_variances = data.var(axis=0, ddof=1)

# Variance of total scores
total_variance = data.sum(axis=1).var(ddof=1)

# Cronbach's alpha formula
alpha = (k / (k - 1)) * (1 - (item_variances.sum() /
total_variance))

return alpha

def cronbachs_alpha_if_deleted(data):
    """Calculate Cronbach's alpha if each item is deleted"""
    results = {}
    for col in data.columns:
        subset = data.drop(columns=[col])
        alpha = cronbachs_alpha(subset)
        results[col] = alpha
    return results

print("=" * 60)
print("CRONBACH'S ALPHA ANALYSIS")
print("=" * 60)

# Calculate Cronbach's alpha for each pair
for pair_name, (col_a, col_b) in zip(['Creativity', 'Authenticity',
'Production Speed', 'Cost Efficiency'], feature_pairs):
    pair_data = dff[[col_a, col_b]]
    alpha = cronbachs_alpha(pair_data)

    print(f"\n{pair_name} Pair:")
    print(f"    Cronbach's Alpha: {alpha:.4f}")

# Interpretation
if alpha >= 0.9:
    interpretation = "Excellent reliability"
elif alpha >= 0.8:
    interpretation = "Good reliability"
elif alpha >= 0.7:
    interpretation = "Acceptable reliability"
elif alpha >= 0.6:
    interpretation = "Questionable reliability"
else:
    interpretation = "Poor reliability"

print(f"    Interpretation: {interpretation}")

# Correlation between items
correlation = pair_data.corr().iloc[0, 1]
print(f"    Inter-item correlation: {correlation:.4f}")

```

```

# Extract only the feature pair columns for overall analysis
feature_columns = [col for pair in feature_pairs for col in pair]
feature_data = dff[feature_columns]

# Overall Cronbach's alpha for all feature pair items
all_alpha = cronbachs_alpha(feature_data)
print(f"\nOverall Cronbach's Alpha (all feature pair items):
{all_alpha:.4f}")

# Alpha if item deleted (only for feature pair columns)
alpha_if_deleted = cronbachs_alpha_if_deleted(feature_data)
print(f"\nCronbach's Alpha if item deleted:")
for item, alpha in alpha_if_deleted.items():
    print(f"    {item}: {alpha:.4f}")

# ===== SHAPIRO-WILK NORMALITY TEST =====
print("\n" + "=" * 60)
print("SHAPIRO-WILK NORMALITY TESTS")
print("=" * 60)

def perform_shapiro_test(data, column_name):
    """Perform Shapiro-Wilk test for normality"""
    statistic, p_value = shapiro(data)

    print(f"\n{column_name}:")
    print(f"    Shapiro-Wilk statistic: {statistic:.4f}")
    print(f"    p-value: {p_value:.6f}")

    if p_value > 0.05:
        print(f"    Result: Data appears normally distributed (p >
0.05)")
    else:
        print(f"    Result: Data does NOT appear normally distributed (p
≤ 0.05)")

    return statistic, p_value

# Test normality for each feature pair variable only
shapiro_results = {}
for col in feature_columns:
    stat, p_val = perform_shapiro_test(dff[col], col)
    shapiro_results[col] = {'statistic': stat, 'p_value': p_val}

# Test normality for differences between pairs (col_b - col_a order)
print(f"\nNormality tests for paired differences:")
difference_normality = {}
for pair_name, (col_a, col_b) in zip(['Creativity', 'Authenticity',
'Production Speed', 'Cost Efficiency'], feature_pairs):

```

```

    diff = dff[col_b] - dff[col_a] # col_b - col_a order
    stat, p_val = perform_shapiro_test(diff, f"{pair_name} difference
({col_b} - {col_a})")
    difference_normality[pair_name] = {'statistic': stat, 'p_value':
p_val, 'normal': p_val > 0.05}

# ===== PAIRED COMPARISONS =====
print("\n" + "=" * 60)
print("PAIRED STATISTICAL TESTS")
print("=" * 60)

def perform_paired_tests(data_b, data_a, pair_name):
    """Perform both paired t-test and Wilcoxon signed-rank test with
col_b - col_a order"""

    print(f"\n{pair_name} Comparison:")
    print(f"  Sample size: {len(data_a)}")
    print(f"  Mean A: {data_a.mean():.3f} (SD: {data_a.std():.3f})")
    print(f"  Mean B: {data_b.mean():.3f} (SD: {data_b.std():.3f})")
    print(f"  Mean difference (B - A): {(data_b -
data_a).mean():.3f}")

    # Paired t-test (comparing B to A, so B - A)
    t_stat, t_p = ttest_rel(data_b, data_a)
    print(f"\n  Paired t-test:")
    print(f"    t-statistic: {t_stat:.4f}")
    print(f"    p-value: {t_p:.6f}")

    # Wilcoxon signed-rank test
    try:
        w_stat, w_p = wilcoxon(data_b, data_a, alternative='two-
sided')
        print(f"\n  Wilcoxon signed-rank test:")
        print(f"    W-statistic: {w_stat:.4f}")
        print(f"    p-value: {w_p:.6f}")
    except ValueError as e:
        print(f"\n  Wilcoxon test error: {e}")
        w_stat, w_p = None, None

    # Effect size (Cohen's d for paired samples) - using B - A
    difference
    diff = data_b - data_a
    cohens_d = diff.mean() / diff.std()
    print(f"\n  Effect size (Cohen's d): {cohens_d:.4f}")

    if abs(cohens_d) < 0.2:
        effect_interpretation = "negligible"
    elif abs(cohens_d) < 0.5:
        effect_interpretation = "small"

```

```

elif abs(cohens_d) < 0.8:
    effect_interpretation = "medium"
else:
    effect_interpretation = "large"

print(f" Effect size interpretation: {effect_interpretation}")

# Interpretation
alpha_level = 0.05
if t_p < alpha_level:
    print(f" Conclusion: Significant difference found (p <
{alpha_level})")
else:
    print(f" Conclusion: No significant difference found (p ≥
{alpha_level})")

return {
    't_stat': t_stat, 't_p': t_p,
    'w_stat': w_stat, 'w_p': w_p,
    'cohens_d': cohens_d,
    'mean_diff': diff.mean()
}

# Perform paired tests with col_b - col_a order
paired_results = {}
for pair_name, (col_a, col_b) in zip(['Creativity', 'Authenticity',
'Production Speed', 'Cost Efficiency'], feature_pairs):
    results = perform_paired_tests(dff[col_b], dff[col_a], pair_name)
    paired_results[pair_name] = results

# ===== CHI-SQUARE TEST =====
print("\n" + "=" * 60)
print("CHI-SQUARE TESTS")
print("=" * 60)

def categorize_scores(data, thresholds=[2, 3, 4, 5]):
    """Categorize continuous scores into ordinal categories"""
    categories = []
    for score in data:
        if score <= thresholds[0]:
            categories.append('Very Low')
        elif score <= thresholds[1]:
            categories.append('Low')
        elif score <= thresholds[2]:
            categories.append('Medium')
        elif score <= thresholds[3]:
            categories.append('High')
        else:
            categories.append('Very High')
    return categories

```

```

def perform_chi_square_test(data_b, data_a, pair_name):
    """Perform chi-square test of independence on categorized data"""

    # Categorize the continuous data
    cat_a = categorize_scores(data_a)
    cat_b = categorize_scores(data_b)

    # Create contingency table
    contingency_df = pd.DataFrame({'A': cat_a, 'B': cat_b})
    contingency_table = pd.crosstab(contingency_df['A'],
    contingency_df['B'])

    print(f"\n{pair_name} Chi-square test:")
    print("Contingency Table:")
    print(contingency_table)

    # Perform chi-square test
    chi2_stat, p_value, dof, expected =
    chi2_contingency(contingency_table)

    print(f"\nChi-square statistic: {chi2_stat:.4f}")
    print(f"Degrees of freedom: {dof}")
    print(f"p-value: {p_value:.6f}")

    # Cramer's V (effect size for chi-square)
    n = contingency_table.sum().sum()
    cramers_v = np.sqrt(chi2_stat / (n * (min(contingency_table.shape)
    - 1)))
    print(f"Cramer's V (effect size): {cramers_v:.4f}")

    if p_value < 0.05:
        print("Result: Significant association between categories (p <
    0.05)")
    else:
        print("Result: No significant association between categories
    (p ≥ 0.05)")

    return {
        'chi2_stat': chi2_stat,
        'p_value': p_value,
        'dof': dof,
        'cramers_v': cramers_v,
        'contingency_table': contingency_table
    }

# Perform chi-square tests with col_b - col_a order
chi_square_results = {}
for pair_name, (col_a, col_b) in zip(['Creativity', 'Authenticity',

```

```

'Production Speed', 'Cost Efficiency'], feature_pairs):
    results = perform_chi_square_test(dff[col_b], dff[col_a],
pair_name)
    chi_square_results[pair_name] = results

# ===== SUMMARY AND RECOMMENDATIONS
=====
print("\n" + "=" * 60)
print("SUMMARY AND RECOMMENDATIONS")
print("=" * 60)

print("\n1. RELIABILITY ASSESSMENT:")
for pair_name, (col_a, col_b) in zip(['Creativity', 'Authenticity',
'Production Speed', 'Cost Efficiency'], feature_pairs):
    pair_data = dff[[col_a, col_b]]
    alpha = cronbachs_alpha(pair_data)
    if alpha >= 0.7:
        print(f"    ✓ {pair_name}: Reliable ( $\alpha$  = {alpha:.3f})")
    else:
        print(f"    △ {pair_name}: Low reliability ( $\alpha$  = {alpha:.3f})")

print("\n2. NORMALITY ASSUMPTIONS:")
normal_pairs = []
for pair_name in ['Creativity', 'Authenticity', 'Production Speed',
'Cost Efficiency']:
    if difference_normality[pair_name]['normal']:
        print(f"    ✓ {pair_name} differences: Normally distributed")
        normal_pairs.append(pair_name)
    else:
        print(f"    △ {pair_name} differences: Not normally
distributed")

print("\n3. STATISTICAL TEST RECOMMENDATIONS:")
for pair_name in ['Creativity', 'Authenticity', 'Production Speed',
'Cost Efficiency']:
    if difference_normality[pair_name]['normal']:
        print(f"    {pair_name}: Use paired t-test (normality
assumption met)")
    else:
        print(f"    {pair_name}: Use Wilcoxon signed-rank test
(normality violated)")

print("\n4. SIGNIFICANT DIFFERENCES FOUND:")
significant_pairs = []
for pair_name in ['Creativity', 'Authenticity', 'Production Speed',
'Cost Efficiency']:
    t_p = paired_results[pair_name]['t_p']
    effect_size = abs(paired_results[pair_name]['cohens_d'])
    mean_diff = paired_results[pair_name]['mean_diff']
    if t_p < 0.05:

```

```

        direction = "B > A" if mean_diff > 0 else "B < A"
        print(f"    ✓ {pair_name}: Significant difference (p = {t_p:.4f}, d = {effect_size:.3f}, {direction})")
        significant_pairs.append(pair_name)
    else:
        print(f"    - {pair_name}: No significant difference (p = {t_p:.4f})")

print(f"\nAnalysis completed for {len(feature_pairs)} feature pairs with {len(dff)} observations.")
print("\nNote: All difference calculations use the order (B - A) where:")
for pair_name, (col_a, col_b) in zip(['Creativity', 'Authenticity', 'Production Speed', 'Cost Efficiency'], feature_pairs):
    print(f"    {pair_name}: {col_b} - {col_a}")

```

# ===== CRONBACH'S ALPHA ANALYSIS =====

## Creativity Pair:

Cronbach's Alpha: 0.1333  
Interpretation: Poor reliability  
Inter-item correlation: 0.0715

## Authenticity Pair:

Cronbach's Alpha: 0.1352  
Interpretation: Poor reliability  
Inter-item correlation: 0.0725

## Production Speed Pair:

Cronbach's Alpha: 0.4590  
Interpretation: Poor reliability  
Inter-item correlation: 0.2985

## Cost Efficiency Pair:

Cronbach's Alpha: 0.2366  
Interpretation: Poor reliability  
Inter-item correlation: 0.1342

Overall Cronbach's Alpha (all feature pair items): 0.5779

## Cronbach's Alpha if item deleted:

creativity\_a: 0.5721  
creativity\_b: 0.5510  
authenticity\_a: 0.5724  
authenticity\_b: 0.5424  
production\_speed\_a: 0.4995  
production\_speed\_b: 0.5259  
cost\_eff\_a: 0.5336

cost\_eff\_b: 0.5564

=====

### SHAPIRO-WILK NORMALITY TESTS

=====

creativity\_a:

Shapiro-Wilk statistic: 0.8863

p-value: 0.000000

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

creativity\_b:

Shapiro-Wilk statistic: 0.8869

p-value: 0.000000

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

authenticity\_a:

Shapiro-Wilk statistic: 0.9400

p-value: 0.000001

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

authenticity\_b:

Shapiro-Wilk statistic: 0.9102

p-value: 0.000000

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

production\_speed\_a:

Shapiro-Wilk statistic: 0.9321

p-value: 0.000000

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

production\_speed\_b:

Shapiro-Wilk statistic: 0.9477

p-value: 0.000004

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

cost\_eff\_a:

Shapiro-Wilk statistic: 0.9241

p-value: 0.000000

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

cost\_eff\_b:

Shapiro-Wilk statistic: 0.9289

p-value: 0.000000

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

Normality tests for paired differences:

Creativity difference (creativity\_b - creativity\_a):

Shapiro-Wilk statistic: 0.9735



p-value: 0.001660

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

Authenticity difference (authenticity\_b - authenticity\_a):

Shapiro-Wilk statistic: 0.9809

p-value: 0.014479

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

Production Speed difference (production\_speed\_b - production\_speed\_a):

Shapiro-Wilk statistic: 0.9708

p-value: 0.000793

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

Cost Efficiency difference (cost\_eff\_b - cost\_eff\_a):

Shapiro-Wilk statistic: 0.9664

p-value: 0.000251

Result: Data does NOT appear normally distributed ( $p \leq 0.05$ )

=====

## PAIRED STATISTICAL TESTS

=====

Creativity Comparison:

Sample size: 180

Mean A: 3.669 (SD: 1.141)

Mean B: 3.731 (SD: 1.094)

Mean difference (B - A): 0.061

Paired t-test:

t-statistic: 0.5383

p-value: 0.591070

Wilcoxon signed-rank test:

W-statistic: 4308.0000

p-value: 0.739210

Effect size (Cohen's d): 0.0401

Effect size interpretation: negligible

Conclusion: No significant difference found ( $p \geq 0.05$ )

Authenticity Comparison:

Sample size: 180

Mean A: 3.542 (SD: 1.041)

Mean B: 3.647 (SD: 1.047)

Mean difference (B - A): 0.106

Paired t-test:

t-statistic: 0.9958

p-value: 0.320668

Wilcoxon signed-rank test:

W-statistic: 4734.0000

p-value: 0.329680

Effect size (Cohen's d): 0.0742

Effect size interpretation: negligible

Conclusion: No significant difference found ( $p \geq 0.05$ )

Production Speed Comparison:

Sample size: 180

Mean A: 3.333 (SD: 1.160)

Mean B: 3.172 (SD: 1.087)

Mean difference (B - A): -0.161

Paired t-test:

t-statistic: -1.6222

p-value: 0.106530

Wilcoxon signed-rank test:

W-statistic: 3961.5000

p-value: 0.074464

Effect size (Cohen's d): -0.1209

Effect size interpretation: negligible

Conclusion: No significant difference found ( $p \geq 0.05$ )

Cost Efficiency Comparison:

Sample size: 180

Mean A: 3.247 (SD: 1.214)

Mean B: 2.981 (SD: 1.222)

Mean difference (B - A): -0.267

Paired t-test:

t-statistic: -2.2325

p-value: 0.026821

Wilcoxon signed-rank test:

W-statistic: 3618.5000

p-value: 0.032043

Effect size (Cohen's d): -0.1664

Effect size interpretation: negligible

Conclusion: Significant difference found ( $p < 0.05$ )

=====

## CHI-SQUARE TESTS

=====

Creativity Chi-square test:

Contingency Table:

B	High	Low	Medium	Very Low
A				
High	35	14	16	9
Low	13	5	10	3
Medium	18	7	15	10
Very Low	11	4	6	4

Chi-square statistic: 4.5717

Degrees of freedom: 9

p-value: 0.869938

Cramer's V (effect size): 0.0920

Result: No significant association between categories ( $p \geq 0.05$ )

Authenticity Chi-square test:

Contingency Table:

B	High	Low	Medium	Very Low
A				
High	23	8	15	5
Low	15	5	25	4
Medium	16	14	15	15
Very Low	7	2	8	3

Chi-square statistic: 18.8001

Degrees of freedom: 9

p-value: 0.026947

Cramer's V (effect size): 0.1866

Result: Significant association between categories ( $p < 0.05$ )

Production Speed Chi-square test:

Contingency Table:

B	High	Low	Medium	Very Low
A				
High	14	11	17	7
Low	7	16	11	10
Medium	6	16	14	8
Very Low	10	6	4	23

Chi-square statistic: 31.6513

Degrees of freedom: 9

p-value: 0.000229

Cramer's V (effect size): 0.2421

Result: Significant association between categories ( $p < 0.05$ )

Cost Efficiency Chi-square test:

Contingency Table:

B	High	Low	Medium	Very Low
A				
High	14	10	7	11
Low	5	11	6	8
Medium	5	7	19	22

Very Low      10      4      14      27

Chi-square statistic: 26.5481

Degrees of freedom: 9

p-value: 0.001661

Cramer's V (effect size): 0.2217

Result: Significant association between categories ( $p < 0.05$ )

=====

## SUMMARY AND RECOMMENDATIONS

=====

### 1. RELIABILITY ASSESSMENT:

- △ Creativity: Low reliability ( $\alpha = 0.133$ )
- △ Authenticity: Low reliability ( $\alpha = 0.135$ )
- △ Production Speed: Low reliability ( $\alpha = 0.459$ )
- △ Cost Efficiency: Low reliability ( $\alpha = 0.237$ )

### 2. NORMALITY ASSUMPTIONS:

- △ Creativity differences: Not normally distributed
- △ Authenticity differences: Not normally distributed
- △ Production Speed differences: Not normally distributed
- △ Cost Efficiency differences: Not normally distributed

### 3. STATISTICAL TEST RECOMMENDATIONS:

Creativity: Use Wilcoxon signed-rank test (normality violated)  
Authenticity: Use Wilcoxon signed-rank test (normality violated)  
Production Speed: Use Wilcoxon signed-rank test (normality violated)  
Cost Efficiency: Use Wilcoxon signed-rank test (normality violated)

### 4. SIGNIFICANT DIFFERENCES FOUND:

- Creativity: No significant difference ( $p = 0.5911$ )
- Authenticity: No significant difference ( $p = 0.3207$ )
- Production Speed: No significant difference ( $p = 0.1065$ )
- ✓ Cost Efficiency: Significant difference ( $p = 0.0268$ ,  $d = 0.166$ ,  $B < A$ )

Analysis completed for 4 feature pairs with 180 observations.

Note: All difference calculations use the order (B - A) where:

Creativity: creativity\_b - creativity\_a

Authenticity: authenticity\_b - authenticity\_a

Production Speed: production\_speed\_b - production\_speed\_a

Cost Efficiency: cost\_eff\_b - cost\_eff\_a

# Using all independent features

```
# Step 1: Define the features and target
features = ['age', 'gender', 'ad_exposure', 'ai_exposure',
'creativity_a', 'authenticity_a', 'production_speed_a', 'cost_eff_a',
'creativity_b', 'authenticity_b', 'production_speed_b', 'cost_eff_b',
'ad_preference']
target_col = 'ad_prediction' # Replace with your actual target column
name

# Step 2: Clean and filter the data
df_model = dff[features + [target_col]].dropna()

# Ensure target is binary (0/1)
df_model = df_model[df_model[target_col].isin([1, 2])]
df_model['target_bin'] = df_model[target_col].apply(lambda x: 1 if x
== 2 else 0)

# Step 3: Add constant for intercept
X = sm.add_constant(df_model[features])
y = df_model['target_bin']

# Step 4: Fit the logistic regression model
logit_model = sm.Logit(y, X).fit()

# Step 5: Display the summary
print(logit_model.summary())

# Step 6: Predict and evaluate
y_pred = (logit_model.predict(X) > 0.5).astype(int)

print("\nConfusion Matrix:")
print(confusion_matrix(y, y_pred))

print("\nClassification Report:")
print(classification_report(y, y_pred, target_names=['Class 1', 'Class
2']))
```

Optimization terminated successfully.

Current function value: 0.225806

Iterations 8

## Logit Regression Results

```
=====
=====
Dep. Variable:          target_bin   No. Observations:
180
Model:                  Logit       Df Residuals:
166
Method:                  MLE        Df Model:
```

13

Date: Sat, 09 Aug 2025 Pseudo R-squ.:

0.6742

Time: 11:06:12 Log-Likelihood:

-40.645

converged: True LL-Null:

-124.76

Covariance Type: nonrobust LLR p-value:

4.244e-29

=====

=====

		coef	std err	z	P> z	
--	--	------	---------	---	------	--

[0.025 0.975]

-----

-----

const		1.4015	2.693	0.520	0.603	-
-------	--	--------	-------	-------	-------	---

3.876 6.679

age		-0.7911	0.404	-1.956	0.050	-
-----	--	---------	-------	--------	-------	---

1.584 0.001

gender		0.6429	0.584	1.101	0.271	-
--------	--	--------	-------	-------	-------	---

0.501 1.787

ad_exposure		0.6072	0.367	1.657	0.098	-
-------------	--	--------	-------	-------	-------	---

0.111 1.326

ai_exposure		0.2783	0.350	0.796	0.426	-
-------------	--	--------	-------	-------	-------	---

0.407 0.964

creativity_a		0.4775	0.453	1.054	0.292	-
--------------	--	--------	-------	-------	-------	---

0.410 1.365

authenticity_a		0.0956	0.467	0.205	0.838	-
----------------	--	--------	-------	-------	-------	---

0.820 1.011

production_speed_a		-0.0392	0.490	-0.080	0.936	-
--------------------	--	---------	-------	--------	-------	---

0.999 0.921

cost_eff_a		-0.6654	0.543	-1.226	0.220	-
------------	--	---------	-------	--------	-------	---

1.729 0.398

creativity_b		0.6057	0.581	1.042	0.297	-
--------------	--	--------	-------	-------	-------	---

0.534 1.745

authenticity_b		-0.9566	0.633	-1.510	0.131	-
----------------	--	---------	-------	--------	-------	---

2.198 0.285

production_speed_b		0.8864	0.455	1.948	0.051	-
--------------------	--	--------	-------	-------	-------	---

0.005 1.778

cost_eff_b		0.3782	0.368	1.027	0.304	-
------------	--	--------	-------	-------	-------	---

0.343 1.100

ad_preference		-4.5739	0.812	-5.636	0.000	-
---------------	--	---------	-------	--------	-------	---

6.165 -2.983

=====

=====

Confusion Matrix:

[[84 7]

[ 8 81]]

### Classification Report:

	precision	recall	f1-score	support
Class 1	0.91	0.92	0.92	91
Class 2	0.92	0.91	0.92	89
accuracy			0.92	180
macro avg	0.92	0.92	0.92	180
weighted avg	0.92	0.92	0.92	180

## Feature selection for model building

dff.corr()

	age	gender	ad_exposure	ai_exposure \
age	1.000000	0.089371	-0.060873	-0.104334
gender	0.089371	1.000000	-0.062697	-0.177550
ad_exposure	-0.060873	-0.062697	1.000000	0.274981
ai_exposure	-0.104334	-0.177550	0.274981	1.000000
creativity_a	-0.095918	-0.077163	0.421676	0.211427
creativity_b	0.020353	0.006805	0.167021	0.265415
authenticity_a	-0.127526	0.033524	0.410958	0.242276
authenticity_b	-0.000623	0.045423	0.159640	0.282053
production_speed_a	0.035644	-0.210512	0.206246	0.213956
production_speed_b	0.111191	-0.107394	0.198930	0.027866
cost_eff_a	0.098918	-0.161750	0.136782	0.179495
cost_eff_b	0.118409	-0.058171	0.159057	0.000818
ad_preference	0.073571	-0.020879	-0.173914	0.060742
ad_prediction	-0.136374	0.010208	0.268101	0.040804

	creativity_a	creativity_b	authenticity_a \
age	-0.095918	0.020353	-0.127526
gender	-0.077163	0.006805	0.033524
ad_exposure	0.421676	0.167021	0.410958
ai_exposure	0.211427	0.265415	0.242276
creativity_a	1.000000	0.071467	0.778320
creativity_b	0.071467	1.000000	0.112953
authenticity_a	0.778320	0.112953	1.000000
authenticity_b	0.026924	0.834539	0.072491
production_speed_a	-0.116733	0.180094	-0.132984
production_speed_b	0.130551	-0.203821	0.084935
cost_eff_a	-0.317778	0.263973	-0.325444
cost_eff_b	0.231789	-0.242218	0.133530
ad_preference	-0.417422	0.399799	-0.371978
ad_prediction	0.502104	-0.300703	0.436678

	authenticity_b	production_speed_a	
production_speed_b \			
age	-0.000623	0.035644	
0.111191			
gender	0.045423	-0.210512	-
0.107394			
ad_exposure	0.159640	0.206246	
0.198930			
ai_exposure	0.282053	0.213956	
0.027866			
creativity_a	0.026924	-0.116733	
0.130551			
creativity_b	0.834539	0.180094	-
0.203821			
authenticity_a	0.072491	-0.132984	
0.084935			
authenticity_b	1.000000	0.242110	-
0.167094			
production_speed_a	0.242110	1.000000	
0.298496			
production_speed_b	-0.167094	0.298496	
1.000000			
cost_eff_a	0.320559	0.771134	
0.261714			
cost_eff_b	-0.248795	0.162231	
0.727914			
ad_preference	0.444157	0.187249	-
0.071725			
ad_prediction	-0.357505	-0.155250	
0.222077			

	cost_eff_a	cost_eff_b	ad_preference	
ad_prediction				
age	0.098918	0.118409	0.073571	-
0.136374				
gender	-0.161750	-0.058171	-0.020879	
0.010208				
ad_exposure	0.136782	0.159057	-0.173914	
0.268101				
ai_exposure	0.179495	0.000818	0.060742	
0.040804				
creativity_a	-0.317778	0.231789	-0.417422	
0.502104				
creativity_b	0.263973	-0.242218	0.399799	-
0.300703				
authenticity_a	-0.325444	0.133530	-0.371978	
0.436678				
authenticity_b	0.320559	-0.248795	0.444157	-
0.357505				



production_speed_a	0.771134	0.162231	0.187249	-
0.155250				
production_speed_b	0.261714	0.727914	-0.071725	
0.222077				
cost_eff_a	1.000000	0.134165	0.364868	-
0.330492				
cost_eff_b	0.134165	1.000000	-0.234837	
0.330440				
ad_preference	0.364868	-0.234837	1.000000	-
0.788938				
ad_prediction	-0.330492	0.330440	-0.788938	
1.000000				

```
dff[['age', 'ad_exposure', 'cost_eff_a', 'cost_eff_b',
      'ad_preference', 'ad_prediction']].corr()
```

	age	ad_exposure	cost_eff_a	cost_eff_b	
ad_preference \					
age	1.000000	-0.060873	0.098918	0.118409	
0.073571					
ad_exposure	-0.060873	1.000000	0.136782	0.159057	-
0.173914					
cost_eff_a	0.098918	0.136782	1.000000	0.134165	
0.364868					
cost_eff_b	0.118409	0.159057	0.134165	1.000000	-
0.234837					
ad_preference	0.073571	-0.173914	0.364868	-0.234837	
1.000000					
ad_prediction	-0.136374	0.268101	-0.330492	0.330440	-
0.788938					

	ad_prediction
age	-0.136374
ad_exposure	0.268101
cost_eff_a	-0.330492
cost_eff_b	0.330440
ad_preference	-0.788938
ad_prediction	1.000000

```
correlation_matrix = dff[['age', 'ad_exposure', 'cost_eff_a',
      'cost_eff_b', 'ad_preference', 'ad_prediction']].corr()
```

```
# Create the heatmap
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix,
```

annot=True,	# Show correlation values
cmap='coolwarm',	# Color scheme (blue-white-red)
center=0,	# Center colormap at 0
square=True,	# Make cells square
fmt='.3f',	# Format numbers to 3 decimal places

```
cbar_kws={'shrink': 0.8}, # Adjust colorbar size
linewidths=0.5)         # Add lines between cells

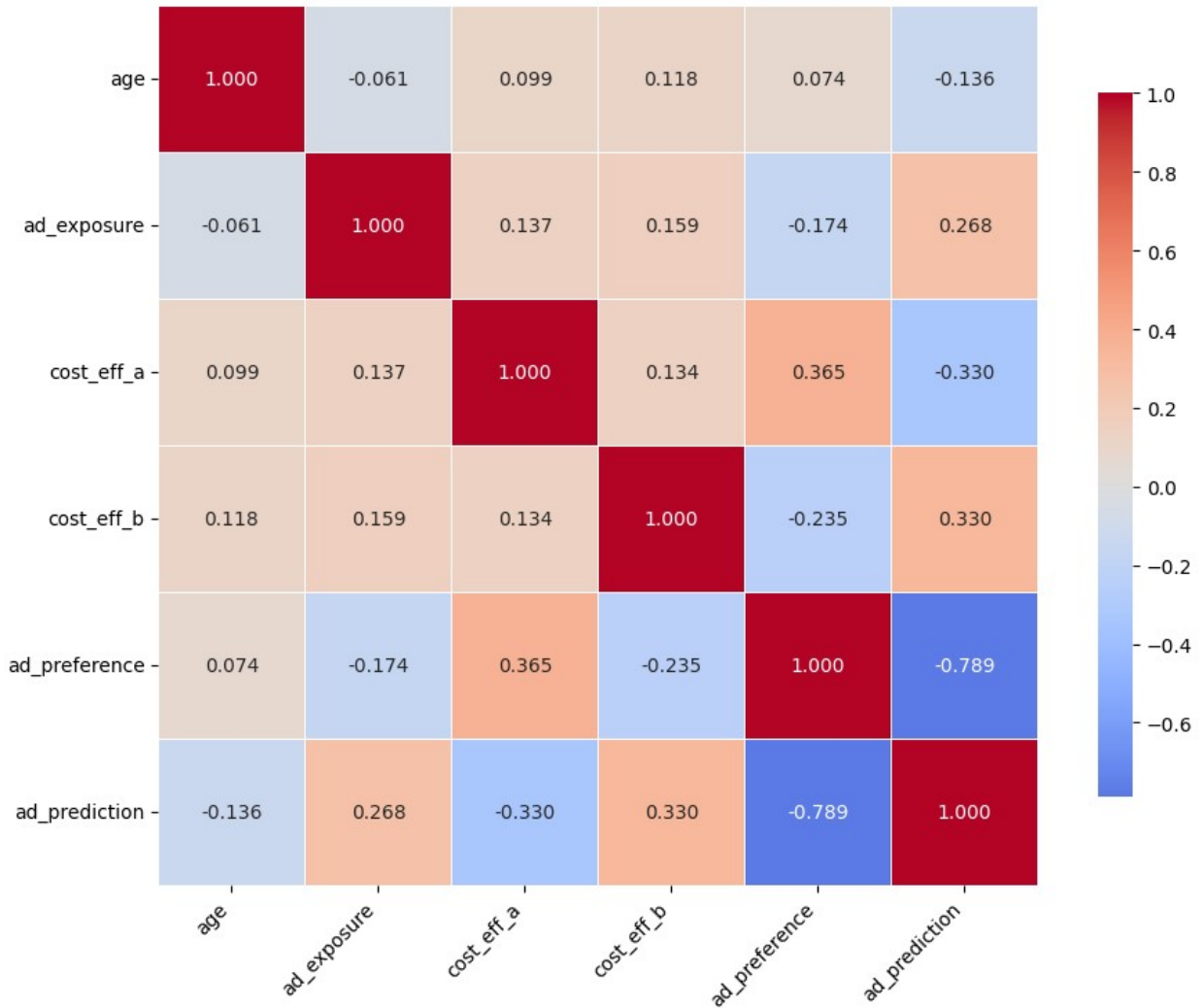
plt.title('Correlation Matrix Heatmap', fontsize=16,
fontweight='bold', pad=20)
plt.xlabel('') # Remove x-axis label (redundant with tick labels)
plt.ylabel('') # Remove y-axis label (redundant with tick labels)
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better
readability
plt.yticks(rotation=0) # Keep y-axis labels horizontal

plt.tight_layout()

plt.show()

# Optional: Print the correlation matrix values
print("Correlation Matrix:")
print(correlation_matrix.round(3))
```

Correlation Matrix Heatmap



Correlation Matrix:

	age	ad_exposure	cost_eff_a	cost_eff_b	
ad_preference \					
age	1.000	-0.061	0.099	0.118	
0.074					
ad_exposure	-0.061	1.000	0.137	0.159	-
0.174					
cost_eff_a	0.099	0.137	1.000	0.134	
0.365					
cost_eff_b	0.118	0.159	0.134	1.000	-
0.235					
ad_preference	0.074	-0.174	0.365	-0.235	
1.000					
ad_prediction	-0.136	0.268	-0.330	0.330	-
0.789					

	ad_prediction
age	-0.136
ad_exposure	0.268
cost_eff_a	-0.330
cost_eff_b	0.330
ad_preference	-0.789
ad_prediction	1.000

## Using only age, ad\_exposure, cost-efficiency and ad preference feature

```
# Step 1: Define the features and target
# features = ['age', 'gender', 'ad_exposure', 'ai_exposure',
# 'cost_eff_a', 'cost_eff_b', 'ad_preference']
features = ['age', 'ad_exposure', 'cost_eff_a', 'cost_eff_b',
'ad_preference']
target_col = 'ad_prediction' # Replace with your actual target column
name

# Step 2: Clean and filter the data
df_model = dff[features + [target_col]].dropna()

# Ensure target is binary (0/1)
df_model = df_model[df_model[target_col].isin([1, 2])]
df_model['target_bin'] = df_model[target_col].apply(lambda x: 1 if x
== 2 else 0)

# Step 3: Add constant for intercept
X = sm.add_constant(df_model[features])
y = df_model['target_bin']

# Step 4: Fit the logistic regression model
logit_model = sm.Logit(y, X).fit()

# Step 5: Display the summary
print(logit_model.summary())

# Step 6: Predict and evaluate
y_pred = (logit_model.predict(X) > 0.5).astype(int)

print("\nConfusion Matrix:")
print(confusion_matrix(y, y_pred))

print("\nClassification Report:")
print(classification_report(y, y_pred, target_names=['Class 1', 'Class
2']))
```

Optimization terminated successfully.  
Current function value: 0.258454  
Iterations 8

### Logit Regression Results

```
=====
=====
Dep. Variable:          target_bin    No. Observations:
180
Model:                  Logit        Df Residuals:
174
Method:                 MLE          Df Model:
5
Date:                   Sat, 09 Aug 2025    Pseudo R-squ.:
0.6271
Time:                   11:06:12    Log-Likelihood:
-46.522
converged:              True          LL-Null:
-124.76
Covariance Type:        nonrobust    LLR p-value:
5.602e-32
=====
=====
```

	coef	std err	z	P> z	[0.025
0.975]					
-----					
-----					
const	4.4950	1.686	2.666	0.008	1.190
7.800					
age	-0.8019	0.359	-2.231	0.026	-1.506
-0.097					
ad_exposure	0.8185	0.309	2.645	0.008	0.212
1.425					
cost_eff_a	-0.7243	0.284	-2.547	0.011	-1.282
-0.167					
cost_eff_b	0.8966	0.261	3.440	0.001	0.386
1.407					
ad_preference	-4.2598	0.614	-6.935	0.000	-5.464
-3.056					

```
=====
=====
Confusion Matrix:
[[84  7]
 [ 9 80]]
```

```
Classification Report:
              precision    recall  f1-score   support

   Class 1         0.90         0.92         0.91         91
```

Class 2	0.92	0.90	0.91	89
accuracy			0.91	180
macro avg	0.91	0.91	0.91	180
weighted avg	0.91	0.91	0.91	180

## Key metrics:

True Negatives (TN): 84 - correctly predicted as class 0 False Positives (FP): 7 - incorrectly predicted as class 1 (Type I error) False Negatives (FN): 9 - incorrectly predicted as class 0 (Type II error) True Positives (TP): 80 - correctly predicted as class 1

## Performance summary:

Total samples: 180 Correct predictions: 164 (84 + 80) Accuracy: 91.1% (164/180) Precision for class 1: 92.0% (80/87) - of predicted positives, how many were correct Recall for class 1: 89.9% (80/89) - of actual positives, how many were caught Precision for class 0: 90.3% (84/93) Recall for class 0: 92.3% (84/91)

## Hypothesis analysis

H1: AI-generated Ads are perceived to be more creative than human-created Ads

Not accepted due to non-significance (No)

H2: AI-generated Ads are perceived to be more authentic than Human-created Ads

Not accepted due to non-significance (No)

H3: AI-generated Ads are perceived to be faster to produce than human-created Ads

Not accepted due to non-significance (No)

H4: AI-generated Ads are perceived to be more cost efficient than human-created Ads

Accepted due to significance (Yes)