In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder,OneHotEncoder,StandardScaler
import warnings
warnings.filterwarnings('ignore')
from sklearn.tree import DecisionTreeRegressor
from sklearn.feature_selection import RFE
from sklearn.preprocessing import MinMaxScaler,StandardScaler
import joblib
import datetime
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.linear_model import Ridge,Lasso
```

In [2]:
```
df= pd.read_csv('advertising.csv')
```

In [3]:
```
df
```

Out[3]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Cli c
0	68.95	35	61833.90	256.09	Cloned 5thgeneration	Wrightburgh	0	Tunisia	2016-03-27	

In [3]:
```
df
```

Out[3]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Cli c
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	
...	...	...	...	...	...	...	...	...	...	
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	2016-02-11 21:49:00	
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	2016-04-22 02:07:01	
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	2016-02-01 17:24:57	
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	2016-03-24 02:35:54	
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	2016-06-03 21:43:21	

1000 rows × 10 columns

In [4]:
```
df.head()
```

Out[4]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	Cloned 5thgeneration	Wrightburgh	0	Tunisia	2016-03-27	0

In [4]:
```
df.head()
```

Out[4]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	0
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	0
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	0
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	0
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	0

In [5]:
```
df.tail()
```

Out[5]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clic on
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	2016-02-11 21:49:00	
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	2016-04-22 02:07:01	
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	2016-02-01 17:24:57	
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	2016-03-24 02:35:54	
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	2016-06-03 21:43:21	

In [6]:
```
df.describe()
```

Out[6]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	65.000200	36.009000	55000.000080	180.000100	0.481000	0.50000
std	15.853615	8.785562	13414.634022	43.902339	0.499889	0.50025

In [6]:
```
df.describe()
```

Out[6]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	65.000200	36.009000	55000.000080	180.000100	0.481000	0.50000
std	15.853615	8.785562	13414.634022	43.902339	0.499889	0.50025
min	32.600000	19.000000	13996.500000	104.780000	0.000000	0.00000
25%	51.360000	29.000000	47031.802500	138.830000	0.000000	0.00000
50%	68.215000	35.000000	57012.300000	183.130000	0.000000	0.50000
75%	78.547500	42.000000	65470.635000	218.792500	1.000000	1.00000
max	91.430000	61.000000	79484.800000	269.960000	1.000000	1.00000

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Daily Time Spent on Site    0
Age                         0
Area Income                 0
Daily Internet Usage        0
Ad Topic Line               0
City                        0
Male                        0
Country                     0
Timestamp                   0
Clicked on Ad               0
dtype: int64
```

In [8]:
```
df.dtypes
```

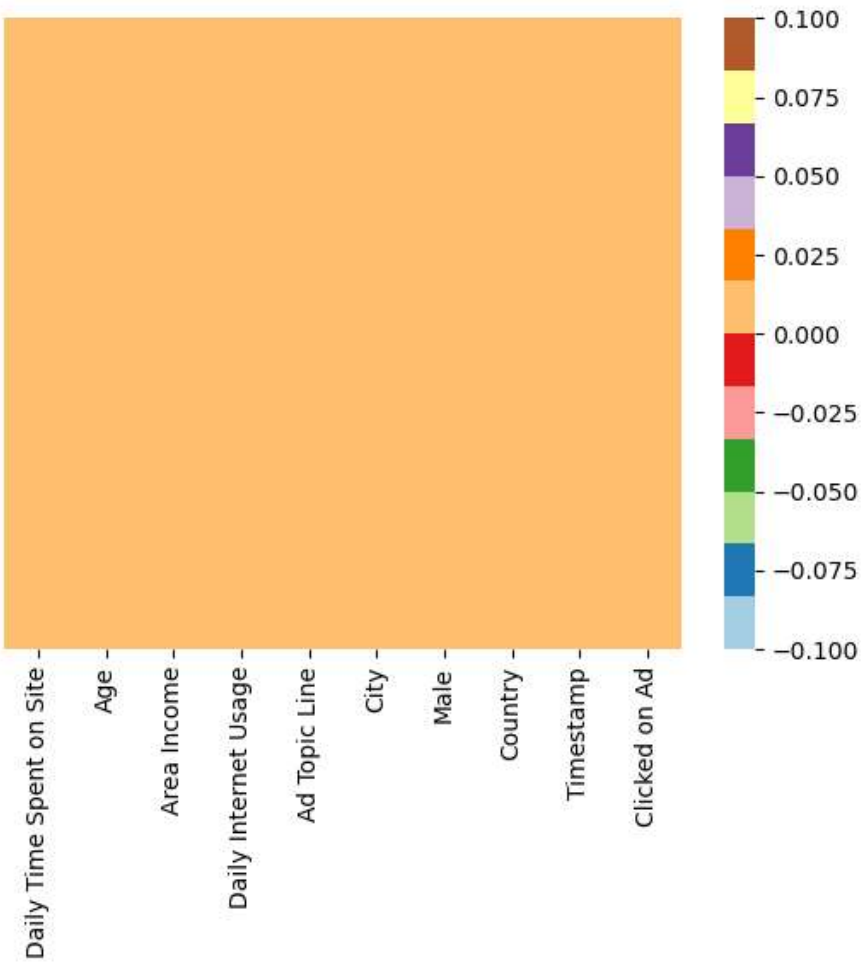
Out[8]:
```
Daily Time Spent on Site    float64
Age                           int64
Area Income                 float64
Daily Internet Usage        float64
Ad Topic Line                object
City                         object
Male                          int64
Country                      object
Timestamp                    object
Clicked on Ad                 int64
dtype: object
```

In [9]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap="Paired")
```

Out[9]:
```
<Axes: >
```

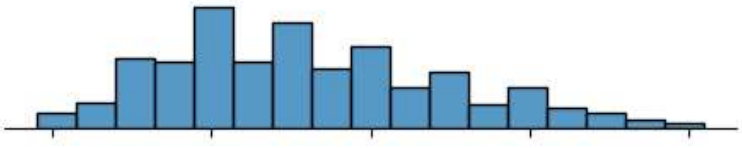
In [9]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap="Paired")
```

Out[9]: <Axes: >



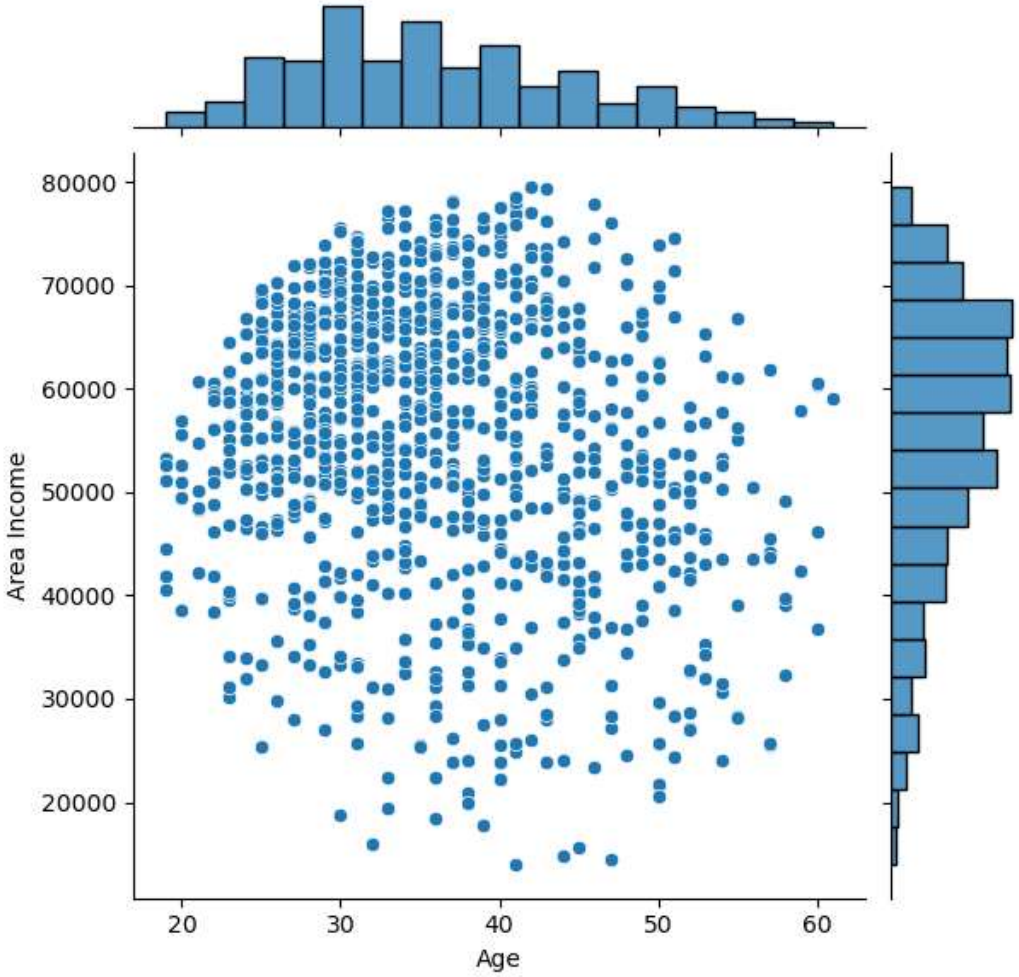
In [10]:
```
sns.jointplot(x='Age',y='Area Income',data=df)
```

Out[10]: <seaborn.axisgrid.JointGrid at 0x2504f763590>

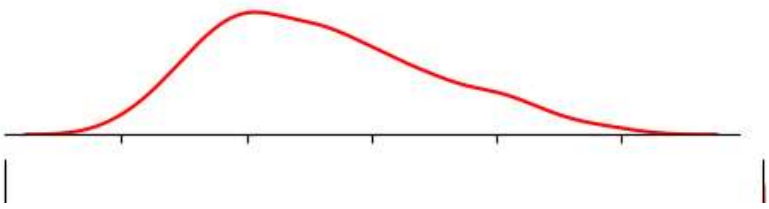


In [10]:
```
sns.jointplot(x='Age',y='Area Income',data=df)
```

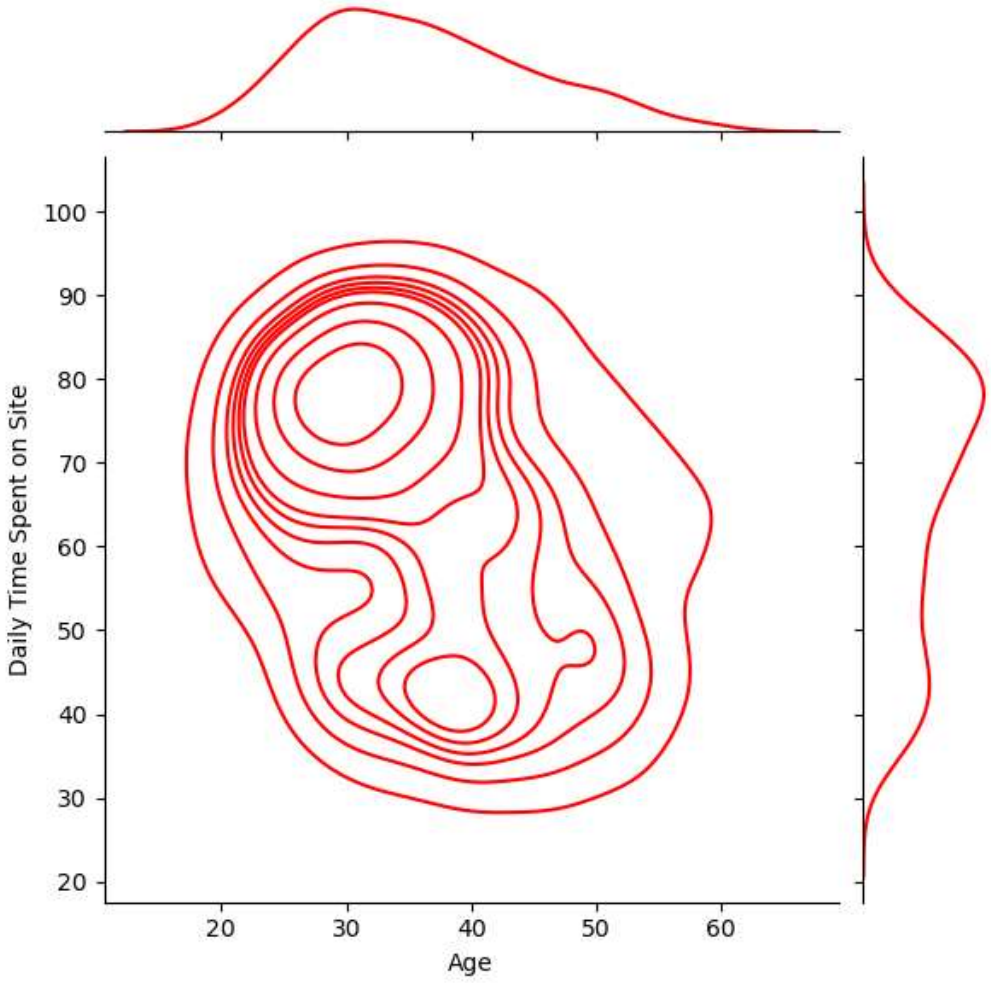
Out[10]:
```
<seaborn.axisgrid.JointGrid at 0x2504f763590>
```



In [11]:
```
sns.jointplot(x='Age',y='Daily Time Spent on Site',data=df,color='red',kind='kde'
```

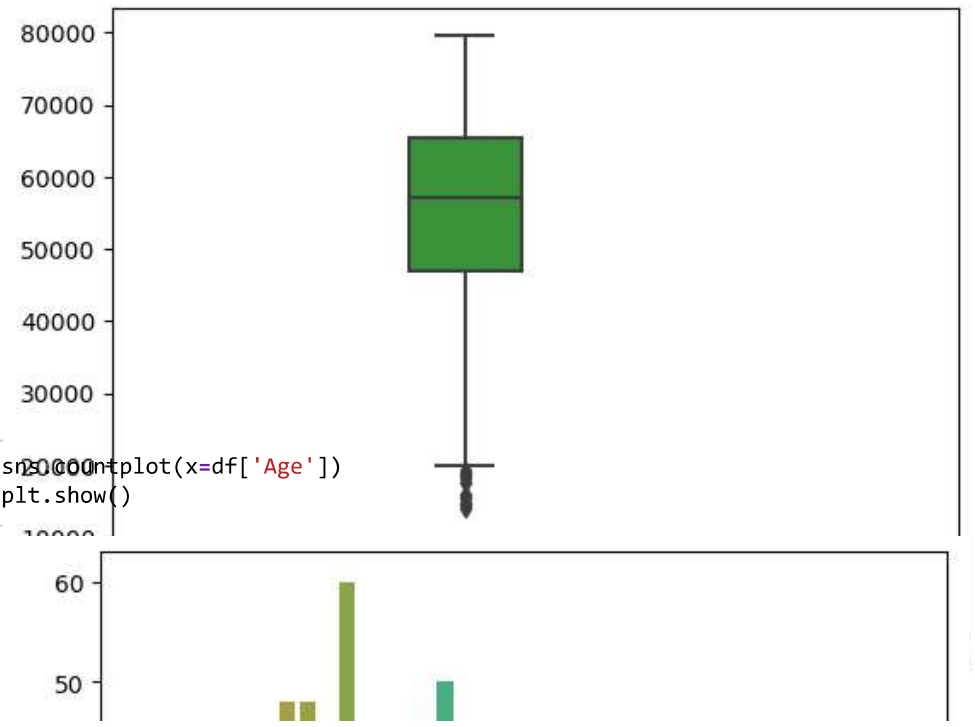


In [11]:
```
sns.jointplot(x='Age',y='Daily Time Spent on Site',data=df,color='red',kind='kde'
```



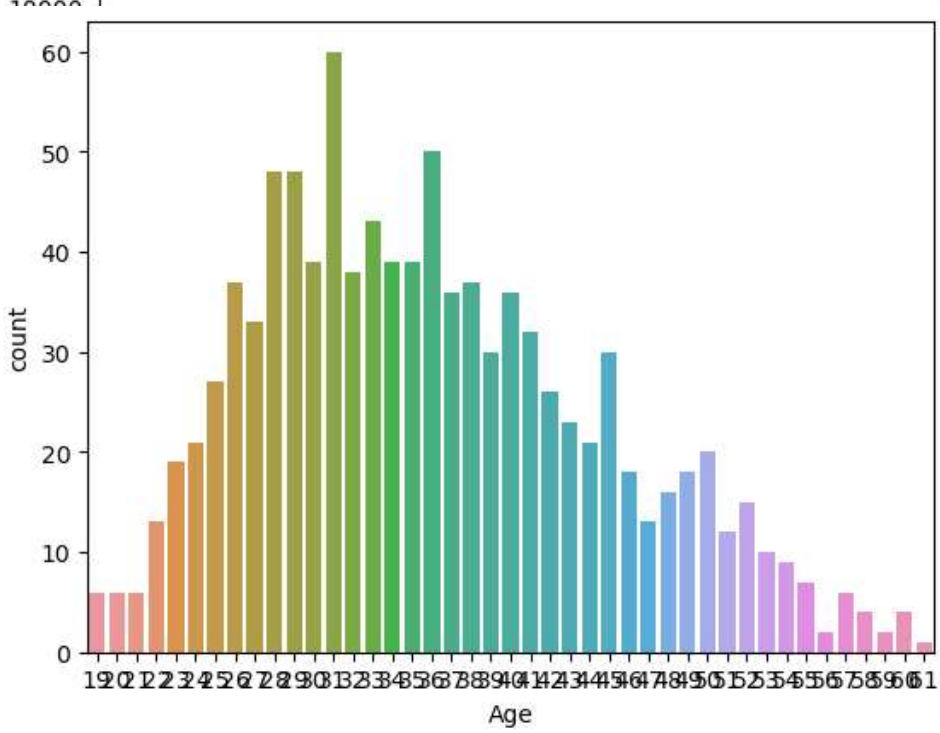
In [12]:
```
sns.boxplot(df)
```

Out[12]: <Axes: >



In [13]:
```
sns.countplot(x=df['Age'])
plt.show()
```

In [13]:
```
sns.countplot(x=df['Age'])
plt.show()
```



In [14]:
```
df.dtypes
```

Out[14]:
```
Daily Time Spent on Site    float64
Age                           int64
Area Income                 float64
Daily Internet Usage        float64
Ad Topic Line                object
City                         object
Male                          int64
Country                      object
Timestamp                    object
Clicked on Ad                 int64
dtype: object
```

In [15]:
```
df
```

Out[15]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Cli c
0	68.95	35	61833.90	256.09	Cloned 5thgeneration	Wrightburgh	0	Tunisia	2016-03-27	

In [15]:
```
df
```

Out[15]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Cli c
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	
...	...	...	...	...	...	...	...	...	...	
995	72.97	30	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	2016-02-11 21:49:00	
996	51.30	45	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	2016-04-22 02:07:01	
997	51.63	51	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	2016-02-01 17:24:57	
998	55.55	19	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	2016-03-24 02:35:54	
999	45.01	26	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	2016-06-03 21:43:21	

1000 rows × 10 columns

In [16]:
```
encoder = LabelEncoder()
```

In [17]:
```
df['Ad Topic Line'] = encoder.fit_transform(df['Ad Topic Line'])
df['City'] = encoder.fit_transform(df['City'])
df['Country'] = encoder.fit_transform(df['Country'])
df['Timestamp'] = encoder.fit_transform(df['Timestamp'])
```

In [18]:
```
df.shape
```

Out[18]:
```
(1000, 10)
```

In [20]:
```
encoder_cols
```

In [19]:
```
categorical_col = ['Ad Topic Line','City','Country','Timestamp']
encoder = OneHotEncoder(drop='first',sparse=False)
encoder_cols = pd.DataFrame(encoder.fit_transform(df[categorical_col]),columns=ar
```

Out[20]:

	Ad Topic Line_1	Ad Topic Line_2	Ad Topic Line_3	Ad Topic Line_4	Ad Topic Line_5	Ad Topic Line_6	Ad Topic Line_7	Ad Topic Line_8	Ad Topic Line_9	Ad Topic Line_10	...
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

Out[18]:
```
(1000, 10)
```

In [20]:
```
encoder_cols
```

In [19]:
```
categorical_col = ['Ad Topic Line','City','Country','Timestamp']
encoder = OneHotEncoder(drop='first',sparse=False)
encoder_cols = pd.DataFrame(encoder.fit_transform(df[categorical_col]),columns=ar
```

Out[20]:

	Ad Topic Line_1	Ad Topic Line_2	Ad Topic Line_3	Ad Topic Line_4	Ad Topic Line_5	Ad Topic Line_6	Ad Topic Line_7	Ad Topic Line_8	Ad Topic Line_9	Ad Topic Line_10	...	Timestam
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

1000 rows × 3202 columns

In [21]:
```
df
```

Out[21]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	91	961	0	215	439	0
1	80.23	31	68441.85	193.77	464	903	1	147	474	0
2	69.47	26	59785.94	236.50	566	111	0	184	367	0
3	74.15	29	54806.18	245.89	903	939	1	103	56	0
4	68.37	35	73889.99	225.58	766	805	0	96	767	0
...	...	...	...	...	...	...	...	...	...	...
995	72.97	30	71384.57	208.58	345	126	1	116	202	1
996	51.30	45	67782.17	134.42	359	488	1	26	567	1
997	51.63	51	42415.72	120.37	263	798	1	140	150	1
998	55.55	19	41920.79	187.95	641	935	0	85	422	0
999	45.01	26	29875.80	178.35	971	744	0	28	772	1

1000 rows × 10 columns

In [22]:
```
numerical_col = ['Daily Time Spent on Site','Area Income','Daily Internet Usage']
Scaled = StandardScaler()
Scaled= pd.DataFrame(Scaled.fit_transform(df[numerical_col]),columns=numerical_co
```

In [23]:
```
Scaled
```

Out[23]:

	Daily Time Spent on Site	Area Income	Daily Internet Usage
0	0.249267	0.509691	1.734030
1	0.961132	1.002530	0.313805
2	0.282083	0.356949	1.287589

In [23]:
```
Scaled
```

Out[23]:

	Daily Time Spent on Site	Area Income	Daily Internet Usage
0	0.249267	0.509691	1.734030
1	0.961132	1.002530	0.313805
2	0.282083	0.356949	1.287589
3	0.577432	-0.014456	1.501580
4	0.212664	1.408868	1.038731
...	...	...	...
995	0.502963	1.222006	0.651314
996	-0.864601	0.953329	-1.038735
997	-0.843775	-0.938570	-1.358924
998	-0.596389	-0.975484	0.181172
999	-1.261555	-1.873832	-0.037605

1000 rows × 3 columns

In [24]:
```
x = pd.concat([encoder_cols,Scaled],axis=1)
y = df['Clicked on Ad']
```

In [25]:
```
x
```

Out[25]:

	Ad Topic Line_1	Ad Topic Line_2	Ad Topic Line_3	Ad Topic Line_4	Ad Topic Line_5	Ad Topic Line_6	Ad Topic Line_7	Ad Topic Line_8	Ad Topic Line_9	Ad Topic Line_10	...	Timestam
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

1000 rows × 3205 columns

In [26]:
```
y
```

Out[26]:
```
0      0
1      0
2      0
3      0
4      0
      ..
995    1
```

In [26]:
```
y
```

Out[26]:
```
0      0
1      0
2      0
3      0
4      0
      ..
995    1
996    1
997    1
998    0
999    1
Name: Clicked on Ad, Length: 1000, dtype: int64
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
```

In [28]:
```
X_train,X_test,Y_train,Y_test= train_test_split(x,y,test_size=0.2,random_state=0)
```

In [29]:
```
log = LogisticRegression()
```

In [30]:
```
y.isnull().sum()
```

Out[30]:
```
0
```

In [31]:
```
log.fit(X_train,Y_train)
```

Out[31]:
```
LogisticRegression
LogisticRegression()
```

In [32]:
```
print ('Train Score:',log.score(X_train,Y_train))
```

```
Train Score: 0.9875
```

In [33]:
```
print('Test Score:',log.score(X_test,Y_test))
```

```
Test Score: 0.935
```

In [34]:
```
pred_train = log.predict(X_train)
pred_test = log.predict(X_test)
```

In [35]:
```
from sklearn import metrics
```

In [36]:
```
print(metrics.classification_report(Y_train,pred_train))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       392
           1       1.00      0.98      0.99       408

    accuracy                           0.99       800
   macro avg       0.99      0.99      0.99       800
weighted avg       0.99      0.99      0.99       800
```

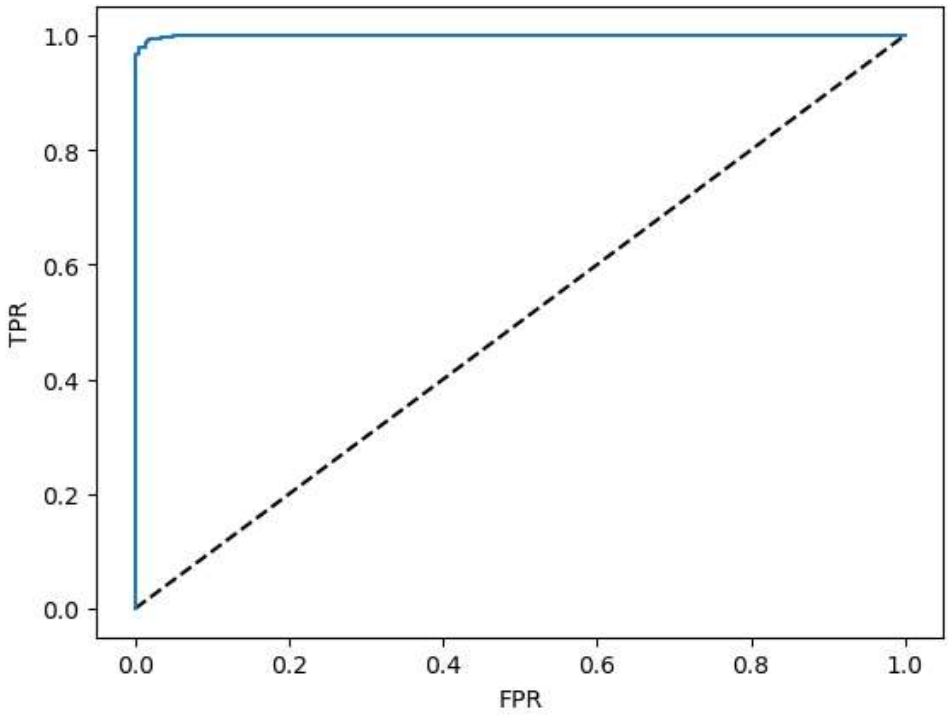
In [37]:
```
df.dtypes
```

Out[37]:
```
Daily Time Spent on Site    float64
Age                           int64
Area Income                 float64
Daily Internet Usage        float64
Ad Topic Line                 int32
City                          int32
Male                          int64
```

In [37]:
```
df.dtypes
```

Out[37]:
```
Daily Time Spent on Site      float64
Age                             int64
Area Income                   float64
Daily Internet Usage          float64
Ad Topic Line                   int32
City                            int32
Male                            int64
Country                         int32
Timestamp                       int32
Clicked on Ad                   int64
dtype: object
```

In [38]:
```
roc = log.predict_proba(X_train)[:,1]
fpr, tpr, threshold = metrics.roc_curve(Y_train, roc)
plt.plot([0,1], [0,1],'k--')
plt.plot(fpr, tpr, label='logistic')
plt.ylabel('TPR')
plt.xlabel('FPR')
plt.show()
```



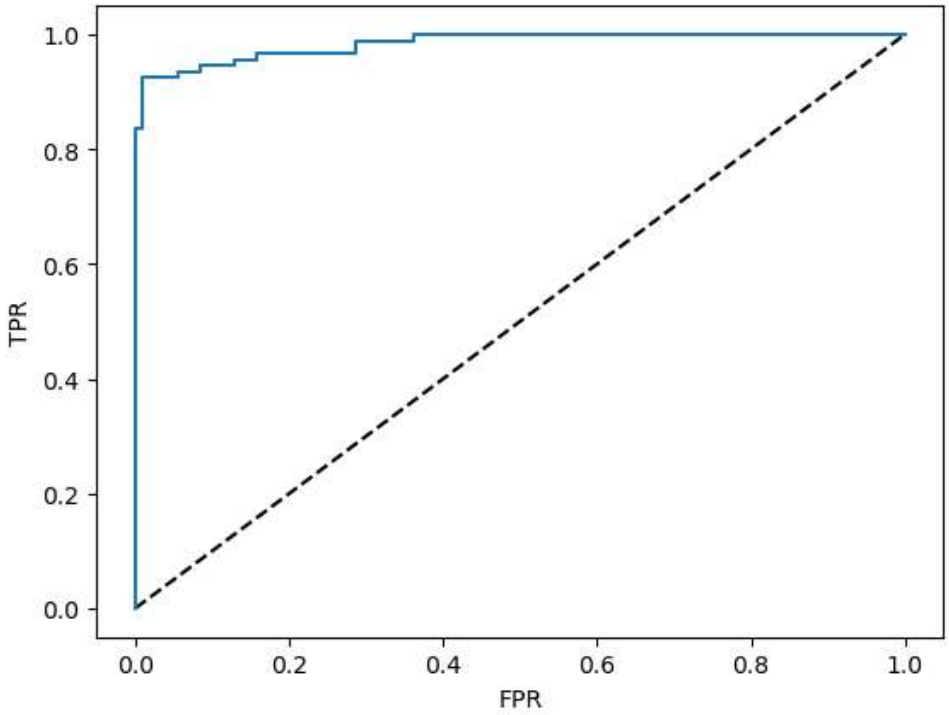
In [39]:
```
metrics.roc_curve(Y_train, roc)
```

Out[39]:
```
(array([0.        , 0.        , 0.        , 0.00255102, 0.00255102,
        0.00510204, 0.00510204, 0.0127551 , 0.0127551 , 0.01530612,
        0.01530612, 0.01785714, 0.01785714, 0.02295918, 0.02295918,
        0.03316327, 0.03316327, 0.04846939, 0.04846939, 1.        ]),
 array([0.        , 0.00245098, 0.96568627, 0.96568627, 0.96813725,
        0.96813725, 0.98039216, 0.98039216, 0.9877451 , 0.9877451 ,
        0.99019608, 0.99019608, 0.99264706, 0.99264706, 0.99509804,
        0.99509804, 0.99754902, 0.99754902, 1.        , 1.        ]),
 array([1.99993446, 0.99993446, 0.67251057, 0.67120995, 0.67092942,
        0.63175682, 0.5095562 , 0.37995648, 0.32607576, 0.31485123,
        0.30375695, 0.30105807, 0.29080246, 0.27585224, 0.27336954,
        0.24990918, 0.24268456, 0.22167799, 0.21972291, 0.00246652]))
```

In [40]:
```
roc = log.predict_proba(X_test)[:,1]
fpr, tpr, threshold = metrics.roc_curve(Y_test, roc)
plt.plot([0,1], [0,1],'k--')
plt.plot(fpr, tpr, label='logistic')
plt.ylabel('TPR')
plt.xlabel('FPR')
plt.show()
```

In [40]:
```
roc = log.predict_proba(X_test)[:,1]
fpr, tpr, threshold = metrics.roc_curve(Y_test, roc)
plt.plot([0,1], [0,1], 'k--')
plt.plot(fpr, tpr, label='logistic')
plt.ylabel('TPR')
plt.xlabel('FPR')
plt.show()
```

```
0.99019608, 0.99019608, 0.99264706, 0.99264706, 0.99509804,
0.99509804, 0.99754902, 0.99754902, 1.        , 1.        ]),
array([1.99993446, 0.99993446, 0.67251057, 0.67120995, 0.67092942,
0.63175682, 0.5095562 , 0.37995648, 0.32607576, 0.31485123,
0.30375695, 0.30105807, 0.29080246, 0.27585224, 0.27336954,
0.24990918, 0.24268456, 0.22167799, 0.21972291, 0.00246652]))
```



In [41]:
```
from sklearn.metrics import matthews_corrcoef

mcc = matthews_corrcoef(Y_test, pred_test)
print('MCC: ',mcc)
```

```
MCC:  0.8691013006857282
```

In [42]:
```
param_grid = {
    'penalty' :['l1','l2'],
    'C' : [0.1,0.5,1,5,10]
}
```

In [43]:
```
from sklearn.model_selection import GridSearchCV
```

In [44]:
```
grid = GridSearchCV(estimator=log, param_grid=param_grid, cv = 5)
```

In [45]:
```
grid.fit(X_train,Y_train)
```

Out[45]:

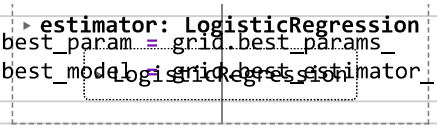
GridSearchCV
estimator: LogisticRegression
LogisticRegression

In [46]:
```
best_param = grid.best_params_
best_model = grid.best_estimator_
```

In [47]:
```
y_pred = best_model.predict(X_test)
```

In [48]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
```

In [46]:
```
best_param = grid.best_params_
best_model = grid.best_estimator_
```



In [47]:
```
y_pred = best_model.predict(X_test)
```

In [48]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
```

In [49]:
```
acc = accuracy_score(Y_test, y_pred)
pre = precision_score(Y_test, y_pred)
rec = recall_score(Y_test, y_pred)
f1 = f1_score(Y_test, y_pred)
roc_auc = roc_auc_score(Y_test, y_pred)
```

In [50]:
```
print('Best Param: ', best_param)
print('Accuracy: ', acc)
print('Recall: ', rec)
print('Precision: ', pre)
print('F1 Score: ', f1)
print('AUC-ROC: ', roc_auc)
```

```
Best Param:  {'C': 0.1, 'penalty': 'l2'}
Accuracy:  0.95
Recall:  0.9239130434782609
Precision:  0.9659090909090909
F1 Score:  0.9444444444444444
AUC-ROC:  0.9480676328502414
```

In [ ]: