In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
raw_data = pd.read_csv('Social_Network_Ads.csv')
raw_data.head()
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [3]:
```
raw_data.describe(include='all')
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
count	4.000000e+02	400	400.000000	400.000000	400.000000
unique	NaN	2	NaN	NaN	NaN
top	NaN	Female	NaN	NaN	NaN
freq	NaN	204	NaN	NaN	NaN
mean	1.569154e+07	NaN	37.655000	69742.500000	0.357500
std	7.165832e+04	NaN	10.482877	34096.960282	0.479864
min	1.556669e+07	NaN	18.000000	15000.000000	0.000000
25%	1.562676e+07	NaN	29.750000	43000.000000	0.000000
50%	1.569434e+07	NaN	37.000000	70000.000000	0.000000
75%	1.575036e+07	NaN	46.000000	88000.000000	1.000000
max	1.581524e+07	NaN	60.000000	150000.000000	1.000000

In [4]:
```
data_no_userid = raw_data.drop(columns=["User ID"], axis = 1)
data_no_userid.describe(include='all')
```

Out[4]:

	Gender	Age	EstimatedSalary	Purchased
count	400	400.000000	400.000000	400.000000
unique	2	NaN	NaN	NaN
top	Female	NaN	NaN	NaN
freq	204	NaN	NaN	NaN
mean	NaN	37.655000	69742.500000	0.357500
std	NaN	10.482877	34096.960282	0.479864
min	NaN	18.000000	15000.000000	0.000000
25%	NaN	29.750000	43000.000000	0.000000
50%	NaN	37.000000	70000.000000	0.000000
75%	NaN	46.000000	88000.000000	1.000000
max	NaN	60.000000	150000.000000	1.000000

In [5]:
```
fig, ax = plt.subplots(figsize = (10, 6))

sns.scatterplot(ax=ax,
                data=data_no_userid,
                x="EstimatedSalary",
                y="Age",
                hue="Purchased")
plt.show()
```

In [ ]:
```

```

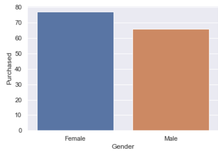
In [7]:
```
data_no_userid["Gender"].unique()
gender = ['Male', 'Female']
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le = le.fit(gender)
data_with_dummies = data_no_userid.copy()
data_with_dummies["Gender"] = le.fit_transform(data_with_dummies["Gender"])
data_with_dummies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Gender           400 non-null    int32
 1   Age              400 non-null    int64
 2   EstimatedSalary  400 non-null    int64
 3   Purchased        400 non-null    int64
dtypes: int32(1), int64(3)
memory usage: 11.1 KB
```

In [ ]:
```

```

In [14]:
```
groupbyGender = pd.DataFrame(data=data_no_userid.groupby(by=["Gender"]).Purchased.sum()).reset_index()
sns.barplot(data=groupbyGender, x="Gender", y="Purchased")
plt.show()
```



In [9]:
```
y = data_with_dummies["Purchased"]
x1 = data_with_dummies.drop(columns=["Purchased"], axis=1)
```

In [10]:
```
x = sm.add_constant(x1)
reg_log = sm.Logit(y,x)
results_log = reg_log.fit()
results_log.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.344804
         Iterations 8
```

Out[10]:

Logit Regression Results

Dep. Variable:	Purchased	**No. Observations:**	400
Model:	Logit	**Df Residuals:**	396
Method:	MLE	**Df Model:**	3
Date:	Wed, 28 Jul 2021	**Pseudo R-squ.:**	0.4711
Time:	10:45:21	**Log-Likelihood:**	-137.92
converged:	True	**LL-Null:**	-260.79
Covariance Type:	nonrobust	**LLR p-value:**	5.488e-53

	coef	std err	z	P>|z|	[0.025	0.975]
const	-12.7836	1.359	-9.405	0.000	-15.448	-10.120
Gender	0.3338	0.305	1.094	0.274	-0.264	0.932
Age	0.2370	0.026	8.984	0.000	0.185	0.289
EstimatedSalary	3.644e-05	5.47e-06	6.659	0.000	2.57e-05	4.72e-05

In [11]:
```
pred_corr = results_log.pred_table()[0, 0] + results_log.pred_table()[1, 1]
pred_incorr = results_log.pred_table()[0, 1] + results_log.pred_table()[1, 0]
total = results_log.pred_table().sum()

accuracy = pred_corr/total*100
print("Accuracy of the model is %.2f" %(accuracy) + '%')
```

```
Accuracy of the model is 85.25%
```

In [12]:
```
y = data_with_dummies["Purchased"]
x1 = data_with_dummies.drop(columns=["Purchased"], axis=1)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
x1_scaled = scaler.fit_transform(x1)
x1_scaled
x = sm.add_constant(x1_scaled)
reg_log = sm.Logit(y,x)
results_log = reg_log.fit()
results_log.summary()
#ACCURACY
pred_corr = results_log.pred_table()[0, 0] + results_log.pred_table()[1, 1]
pred_incorr = results_log.pred_table()[0, 1] + results_log.pred_table()[1, 0]
total = results_log.pred_table().sum()

accuracy = pred_corr/total*100
print("Accuracy of the model is %.2f" %(accuracy) + '%')
```

```
Optimization terminated successfully.
         Current function value: 0.344804
         Iterations 8
Accuracy of the model is 85.25%
```

In [13]:
```
x_train, x_test, y_train, y_test = train_test_split(x1_scaled, y, test_size=0.2, random_state = 42)

model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
y_pred
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
pred_corr = cm[0, 0] + cm[1, 1]
pred_incorr = cm[0, 1] + cm[1, 0]
total = cm.sum()

accuracy = pred_corr/total*100
print("Accuracy of the model is %.2f" %(accuracy) + '%')
acc_score = accuracy_score(y_test, y_pred)
print("Accuracy score is %.2f" %(acc_score*100)+ '%')
classification_report(y_test, y_pred)
```

```
Accuracy of the model is 87.50%
Accuracy score is 87.50%
```

Out[13]:
```
'              precision    recall  f1-score   support\n\n           0       0.85      0.98
0.91        52\n           1       0.95      0.68      0.79        28\n\n    accuracy
0.88        80\n   macro avg       0.90      0.83      0.85        80\nweighted avg       0.8
9      0.88      0.87        80\n'
```

In [ ]:
```

```