Python x Machine Learning Algorithms

Training & visualizing machine learning models against marketing dataset to predict campaign offer acceptance

Natalie Cheng
Data Science Certificate, Noble Desktop (2023)

The Dataset

The public dataset ‘Marketing Campaign’ was downloaded from Kaggle for purposes of understanding if & how different factors would influence a customer’s decision to accept an offer from the campaign. The initial expectation was that annual household income could be a strong candidate to help predict whether a customer would accept an offer, however the dataset offered additional factors that were also considered & evaluated.

The selected machine learning models were employed in various capacities, as were charts for visualization. It is important to remember that while these models were cleaned & trained as effectively as possible, there are still external factors to consider when leveraging these insights for market research.

https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign?select=marketing_campaign.csv

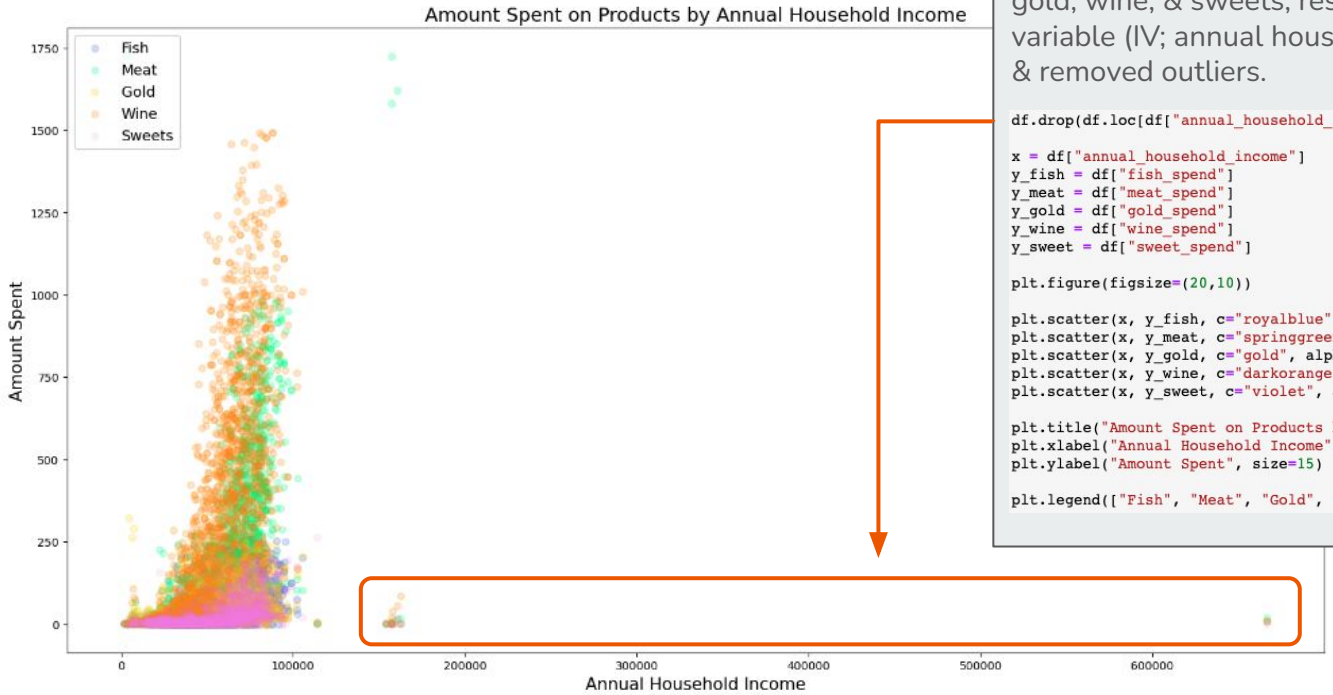
Data Cleaning

Imported necessary libraries & marketing dataset as new DataFrame ‘df’. Reviewed missing values & decided to drop 24 rows of missing data. Dropped unnecessary columns & renamed columns for analysis.

```
df = pd.read_excel("/Users/npcheng/Downloads/marketing_campaign.xlsx")
df.dropna(inplace=True)
df.drop(columns=["ID","Dt_Customer","Z_CostContact","Z_Revenue","Complain"], inplace=True)

df.rename(columns={"Year_Birth":"birth_year",
                   "Education":"education_level",
                   "Marital_Status":"marital_status",
                   "Income":"annual_household_income",
                   "Kidhome":"num_kids",
                   "Teenhome":"num_teens",
                   "Recency":"days_since_last_purchase",
                   "MntWines":"wine_spend",
                   "MntFruits":"fruit_spend",
                   "MntMeatProducts":"meat_spend",
                   "MntFishProducts":"fish_spend",
                   "MntSweetProducts":"sweet_spend",
                   "MntGoldProds":"gold_spend",
                   "NumDealsPurchases":"purchases_w_discount",
                   "NumWebPurchases":"purchases_thru_website",
                   "NumCatalogPurchases":"purchases_thru_catalog",
                   "NumStorePurchases":"purchases_in_store",
                   "NumWebVisitsMonth":"web_visits_last_month",
                   "AcceptedCmp3":"accepted_offer3",
                   "AcceptedCmp4":"accepted_offer4",
                   "AcceptedCmp5":"accepted_offer5",
                   "AcceptedCmp1":"accepted_offer1",
                   "AcceptedCmp2":"accepted_offer2",
                   "Response":"accepted_offer"}, inplace=True)
```

What is the relationship between annual household income & amount spent on various products?

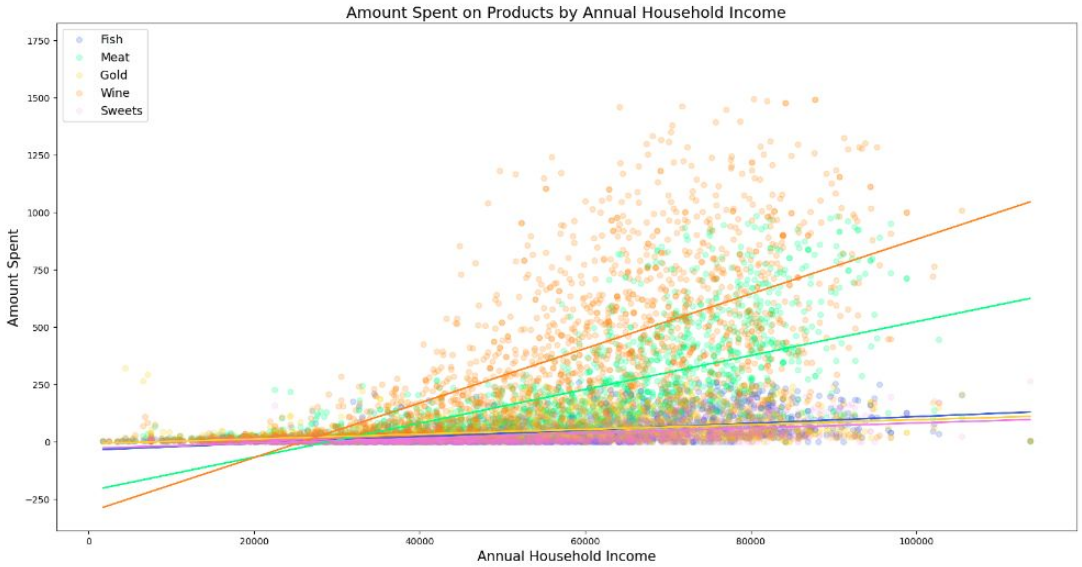
Linear Regression Data Exploration



Plotted dependent variables (DV; amount spent on fish, meat, gold, wine, & sweets, respectively) against independent variable (IV; annual household income) as a scatter. Reviewed & removed outliers.

```
df.drop(df.loc[df["annual_household_income"] > 120000].index, inplace=True)

x = df["annual_household_income"]
y_fish = df["fish_spend"]
y_meat = df["meat_spend"]
y_gold = df["gold_spend"]
y_wine = df["wine_spend"]
y_sweet = df["sweet_spend"]

plt.figure(figsize=(20,10))

plt.scatter(x, y_fish, c="royalblue", alpha=0.2)
plt.scatter(x, y_meat, c="springgreen", alpha=0.2)
plt.scatter(x, y_gold, c="gold", alpha=0.2)
plt.scatter(x, y_wine, c="darkorange", alpha=0.2)
plt.scatter(x, y_sweet, c="violet", alpha=0.1)

plt.title("Amount Spent on Products by Annual Household Income", size=17)
plt.xlabel("Annual Household Income", size=15)
plt.ylabel("Amount Spent", size=15)

plt.legend(["Fish", "Meat", "Gold", "Wine", "Sweets"], ncol=1, loc="upper left", fontsize=13)
```

Linear Regression Single IV



As annual household income increases, the sample audience is most likely to increase spending on wine, followed by meat. The data showed that increases in annual household income did not correlate with significant increases in spend on fish, gold, or sweets.

Set values for IV (X) & DVs (y_f, y_m, y_g y_w, y_s) to train models.

```
X = df[["annual_household_income"]]
y_f = df["fish_spend"]
y_m = df["meat_spend"]
y_g = df["gold_spend"]
y_w = df["wine_spend"]
y_s = df["sweet_spend"]
```

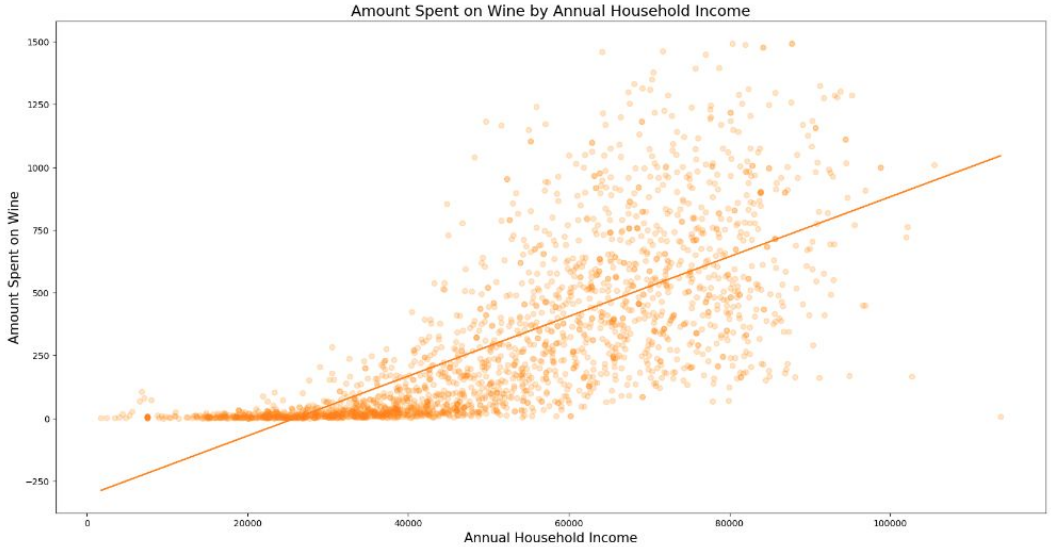
Initiated & trained Linear Regression models for each DV against IV.

```
model_fish = linear_model.LinearRegression()
model_fish.fit(X, y_f)

model_meat = linear_model.LinearRegression()
model_meat.fit(X, y_m)

model_gold = linear_model.LinearRegression()
model_gold.fit(X, y_g)

model_wine = linear_model.LinearRegression()
model_wine.fit(X, y_w)

model_sweet = linear_model.LinearRegression()
model_sweet.fit(X, y_s)
```

Plotted models against scatter to help predict amount spent on various products based on annual household income.

```
plt.plot(df["annual_household_income"], model_fish.predict(X), c="royalblue")
plt.plot(df["annual_household_income"], model_meat.predict(X), c="springgreen")
plt.plot(df["annual_household_income"], model_gold.predict(X), c="gold")
plt.plot(df["annual_household_income"], model_wine.predict(X), c="darkorange")
plt.plot(df["annual_household_income"], model_sweet.predict(X), c="violet")
```

Linear Regression Single IV



Focusing on the Linear Regression model for amount spent on wine by annual household income. Ran train_test_split to evaluate model score.

```
1 X = df[["annual_household_income"]]
2 y = df["wine_spend"]
```

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

```
1 model = linear_model.LinearRegression()
2 model.fit(X_train, y_train)
```

▾ LinearRegression

LinearRegression()

```
1 model.score(X_test, y_test)
```

```
0.5484909990555072
```

Model returned ~55% accuracy.

Linear Regression Single IV

Created new DataFrame 'df_predicted' to compare predicted values against actual results.

```
predicted = model.predict(X_test)

df_predicted = pd.DataFrame()
df_predicted["Actual"] = y_test.values
df_predicted["Predicted"] = predicted
```

Then concatenated df_predicted with test values & outputted first 15 rows for comparison. Examples of relatively accurate (green) & inaccurate (red) predictions outlined.

```
X_test.reset_index(drop=True, inplace=True)
pd.concat([X_test, df_predicted], axis=1).head(15)
```

	annual_household_income	Actual	Predicted
0	89058.0	454	757.464546
1	42386.0	65	194.543729
2	90226.0	1083	771.552039
3	82584.0	1076	679.380269
4	12571.0	3	-165.061262
5	73059.0	410	564.497239
6	41443.0	171	183.170008
7	7500.0	5	-226.223660
8	45938.0	273	237.385149
9	82427.0	482	677.486659
10	38823.0	70	151.569636
11	48186.0	97	264.498750
12	58116.0	228	384.266570
13	24683.0	8	-18.975880
14	49572.0	35	281.215588

Linear Regression Multiple IVs, Lasso, Ridge

```
lasso = linear_model.Lasso(alpha=1.0)
lasso.fit(X_train, y_train)
```

Lasso()

```
ridge = linear_model.Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
```

Ridge()

Initiated & trained Lasso & Ridge models to compare against Linear Regression model.

Then introduced additional IVs (all_X) & StandardScaler to help normalize data & prevent certain features from dominating algorithm. Ran train_test_split on new Linear Regression, Lasso, & Ridge models & outputted scores via a for loop.

Final output reflects differences in model scores when multiple IVs are passed in, with Linear & Ridge being nearly identical.

```
all_X = df[["annual_household_income",
            "num_kids",
            "num_teens",
            "days_since_last_purchase",
            "web_visits_last_month",
            "fruit_spend",
            "meat_spend",
            "sweet_spend",
            "gold_spend"]]
y = df["wine_spend"]

X_train, X_test, y_train, y_test = train_test_split(all_X, y, test_size=0.2)
```

```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
models = (("Linear", linear_model.LinearRegression()),
          ("Lasso", linear_model.Lasso()),
          ("Ridge", linear_model.Ridge()))
```

```
for model_name, model in models:
    model.fit(X_train, y_train)
    print(model_name, f"Score: {round(model.score(X_test,y_test), 4)}")
```

```
Linear Score: 0.604
Lasso Score: 0.6046
Ridge Score: 0.6041
```

Are there any notable correlations between marital status & annual household income, wine spend, or meat spend?

Correlations Pairplot

Cleaned ‘marital_status’ data in df to remove extraneous responses & focus on top five values.

```
df["marital_status"].value_counts()
```

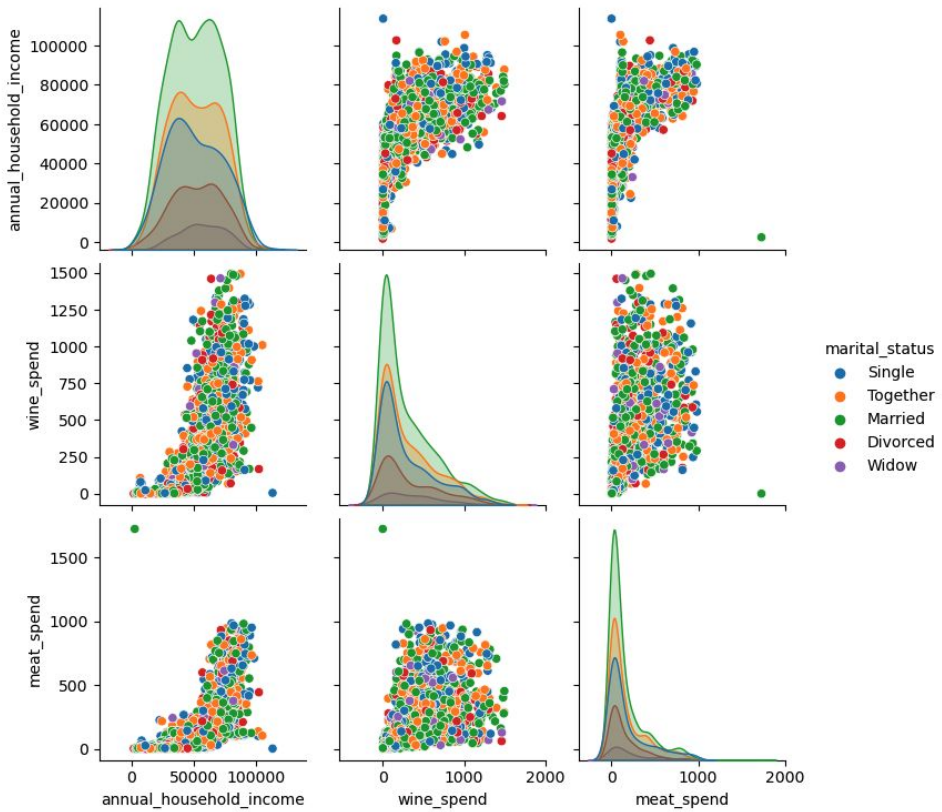
```
Married     854
Together    569
Single      471
Divorced    231
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: marital_status, dtype: int64
```

```
drop_list = ["Alone", "Absurd", "YOLO"]
df.drop(df.query("marital_status in @drop_list").index, inplace=True)
df["marital_status"].unique()
```

```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow'], dtype=object)
```

Then created new DataFrame ‘df_pairplot’ to plot relationships between marital status, annual household income, & spend on wine & meat (as these were identified to be more strongly influenced by annual household income than other forms of spend).

```
df_pairplot = pd.DataFrame()
df_pairplot = df[["marital_status","annual_household_income", "wine_spend", "meat_spend"]]
sns.pairplot(data=df_pairplot, hue="marital_status")
```

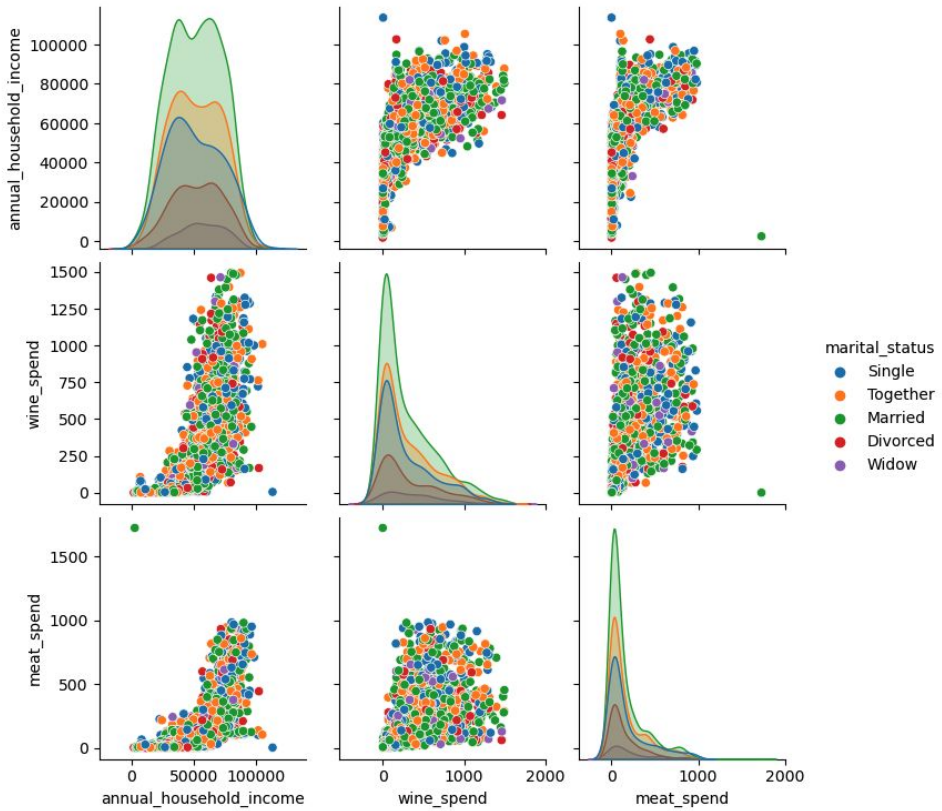


Correlations Pairplot

The pairplot shows that annual household income is highest for customers with a marital status of married, followed in order by together, single, divorced, & widowed.

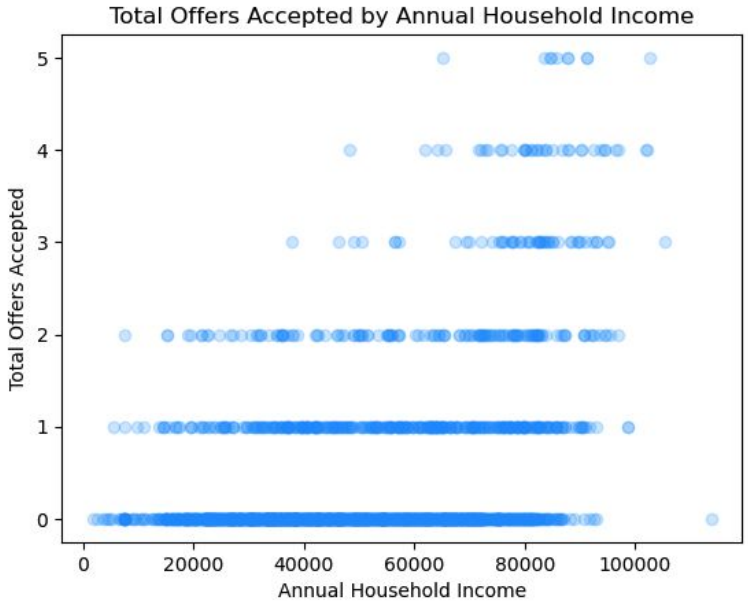
The plot also reflects a notable increase in wine spend across all marital statuses as annual household income increases, as well as a slight increase in meat spend.

At a high level, wine spend & meat spend do not appear to show any significant correlations, implying that changes in spend to one area are unlikely to predict changes in spend to the other.



Can annual household income help predict whether a customer will or will not accept an offer from a marketing campaign?

Logistic Regression Data Exploration



Created new column 'total_offers_accepted' by adding binary values (0 = offer not accepted, 1 = offer accepted) from each offer column.

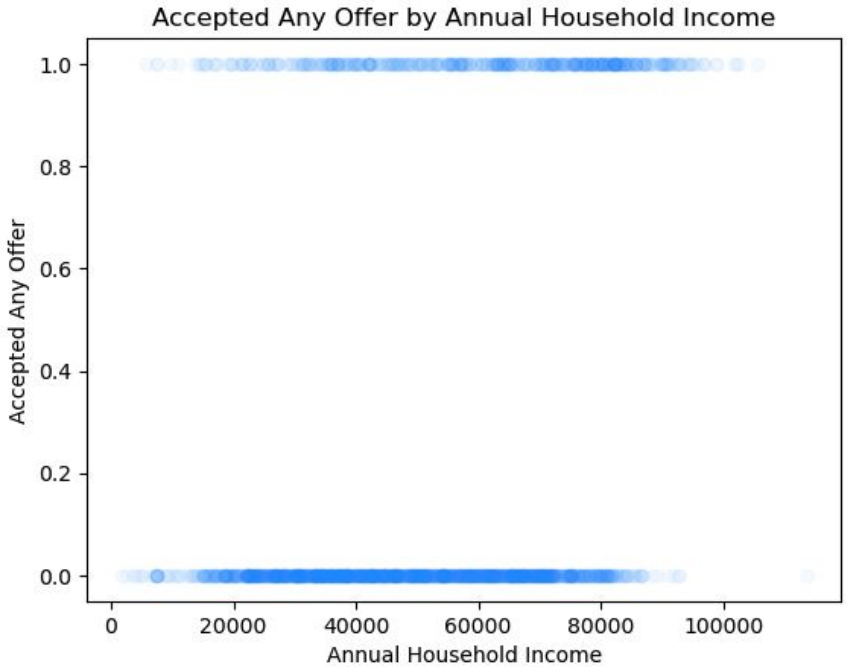
```
df["total_offers_accepted"] = df["accepted_offer1"] +
                              df["accepted_offer2"] +
                              df["accepted_offer3"] +
                              df["accepted_offer4"] +
                              df["accepted_offer5"] +
                              df["accepted_offer"]
```

Plotted total_offers_accepted against annual household income to visualize relationships between income & offers accepted.

```
x = df["annual_household_income"]
y = df["total_offers_accepted"]

plt.scatter(x, y, c="dodgerblue", alpha=0.2)

plt.title("Total Offers Accepted by Annual Household Income")
plt.xlabel("Annual Household Income")
plt.ylabel("Total Offers Accepted")
```

At a high level, customers with higher annual household incomes accepted more offers more frequently than customers with lower annual household incomes.

Logistic Regression



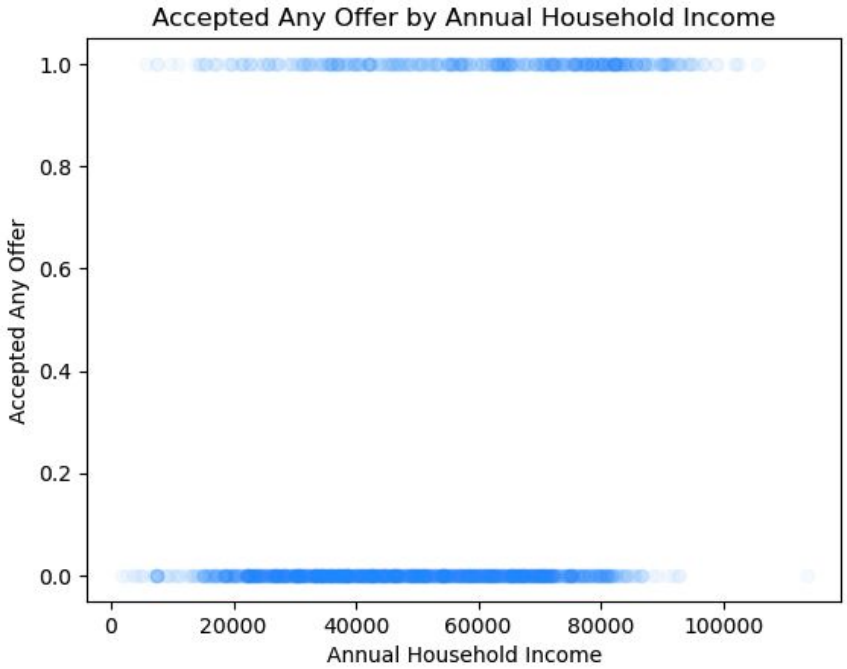
Created function ‘any_offer’ to populate new column ‘accepted_any_offer’ with binary values 0 (no offers accepted) & 1 (at least one offer accepted).

```
def any_offer(row):
    if row["total_offers_accepted"] > 0:
        return 1
    else:
        return 0

df["accepted_any_offer"] = df.apply(lambda row: any_offer(row), axis=1)
```

Plotted accepted_any_offer against annual household income to view relationship between annual household income & accepting any offer.

```
x = df["annual_household_income"]
y = df["accepted_any_offer"]

plt.scatter(x, y, c="dodgerblue", alpha=0.03)

plt.title("Accepted Any Offer by Annual Household Income")
plt.xlabel("Annual Household Income")
plt.ylabel("Accepted Any Offer")
```

Logistic Regression



Initiated & trained Logistic Regression model to predict whether any offer would be accepted (0 = no, 1 = yes) based on inputted annual household income.

```
X = df[["annual_household_income"]]
y = df["accepted_any_offer"]
```

```
model = linear_model.LogisticRegression()
model.fit(X, y)
```

Used model to predict offer acceptance for annual household incomes of $60,000 & $100,000.

```
model.predict([[60000]])
```

```
/Users/npcheng/anaconda3/lib/p
names, but LogisticRegression
  warnings.warn(
```

```
array([0])
```

```
model.predict([[100000]])
```

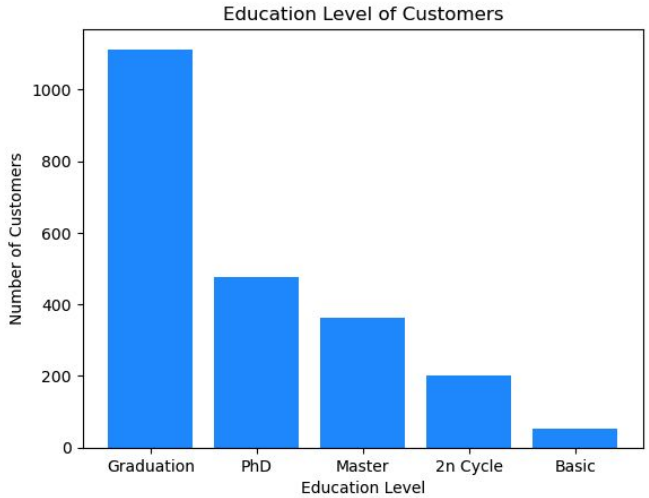
```
/Users/npcheng/anaconda3/lib/p
names, but LogisticRegression
  warnings.warn(
```

```
array([1])
```

The Logistic Regression model predicted that a customer with an annual household income of $60,000 would not accept any offer, while a customer with an annual household income of $100,000 would accept at least one offer. However, due to the high variance in data, this model is likely not a strong candidate for accurately & consistently predicting when a customer would accept an offer based on annual household income.

Can level of education be a considerable factor in predicting whether a customer will accept any offer?

Logistic Regression get_dummies



Plotted education values as a bar to visualize number of customers with each level of education.

```
plt.bar(education.index, education.values, color="dodgerblue")
plt.title("Education Level of Customers")
plt.xlabel("Education Level")
plt.ylabel("Number of Customers")
```

Then created new DataFrame 'edu_dummies' & converted categorical values (Graduation, PhD, Master, 2n Cycle, Basic) into indicator values with get_dummies. Concatenated edu_dummies with df.

```
edu_dummies = pd.get_dummies(df["education_level"])
df = pd.concat([df, edu_dummies], axis=1)
```

```
1 df.head()
```

fruit_spend	meat_spend	...	revenue_after_offer	accepted_offer	total_offers_accepted	accepted_any_offer	total_children	2n Cycle	Basic	Graduation	Master	PhD
88	546	...	11	1	1	1	0	0	0	1	0	0
1	6	...	11	0	0	0	2	0	0	1	0	0
49	127	...	11	0	0	0	0	0	0	1	0	0
4	20	...	11	0	0	0	1	0	0	1	0	0
43	118	...	11	0	0	0	1	0	0	0	0	1

Logistic Regression Confusion Matrix

Established IVs (X; education levels, annual_household_income, total_children) & DV (y; accepted_any_offer), then ran train_test_split to train Logistic Regression model & output model score.

```
X = df[["2n Cycle", "Basic", "Graduation", "Master", "PhD", "annual_household_income", "total_children"]]
y = df["accepted_any_offer"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
model_logistic = linear_model.LogisticRegression()
```

```
model_logistic.fit(X_train, y_train)
```

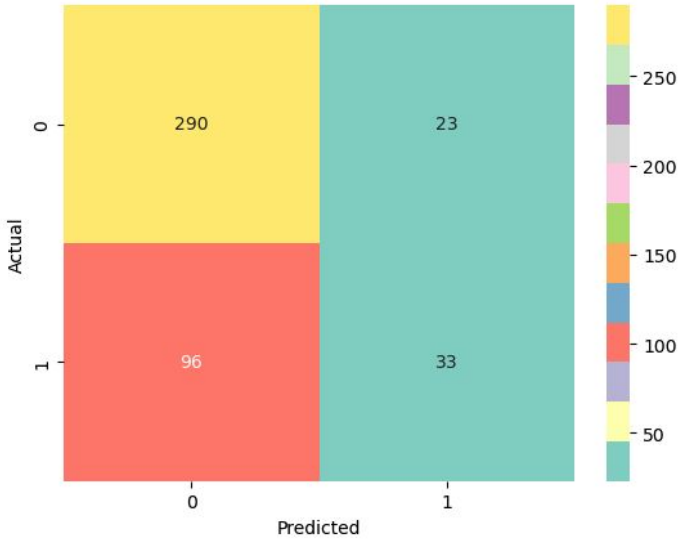
```
LogisticRegression()
```

```
model_logistic.score(X_test, y_test)
```

```
0.7307692307692307
```

Model returned ~73% accuracy.

Created a Confusion Matrix to compare actual values against predicted values.



The Confusion Matrix shows that the model was fairly successful in predicting 0 (did not accept any offers), accurately predicting 290 out of 313 results. The model was less successful in predicting 1 (accepted any offer), accurately predicting 33 out of 129 results.

KNN (K-Nearest Neighbor)

Using the same data as the Logistic Regression model, initiated & trained a KNN model, then tested it.

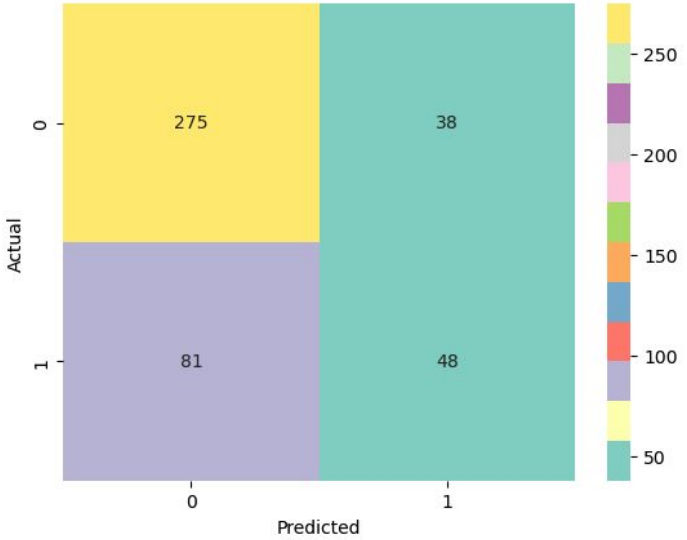
```
model_knn = KNeighborsClassifier(n_neighbors=5)
model_knn.fit(X_train, y_train)
model_knn.score(X_test, y_test)
```

```
0.7307692307692307
```

Similar to the Logistic Regression model, the KNN model returned ~73% accuracy.

Created a Confusion Matrix to compare actual values against predicted values.

The Confusion Matrix shows that the model was fairly successful in predicting 0 (did not accept any offers), accurately predicting 275 out of 313 results. The model was less successful in predicting 1 (accepted any offer), accurately predicting 48 out of 129 results. Compared to the Logistic Regression model, the KNN model performed slightly worse in predicting 0 & slightly better in predicting 1.



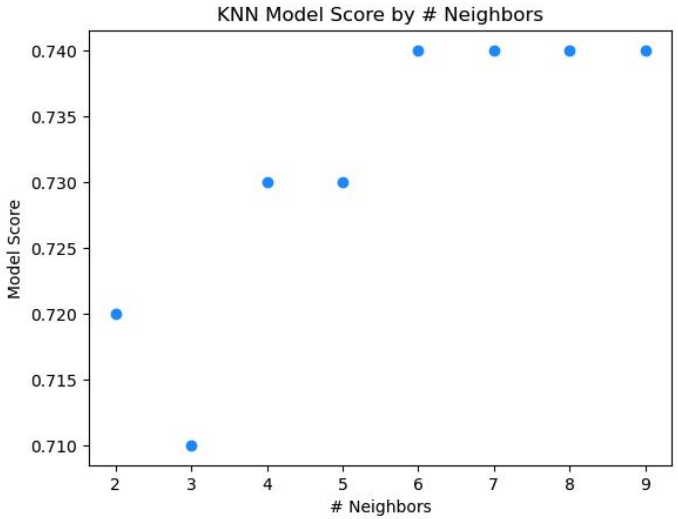
KNN (K-Nearest Neighbor) For Loop

Ran a for loop to understand how the KNN model changed with fewer or additional neighbors (original = 5).

```
n_neighbors = []
scores = []

for i in range(2,10):

    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train, y_train)

    score = round(model.score(X_test, y_test), 2)
    print(f"# Neighbors: {i} | Model Score: {score}")

    n_neighbors.append(i)
    scores.append(score)
```

```
# Neighbors: 2 | Model Score: 0.72
# Neighbors: 3 | Model Score: 0.71
# Neighbors: 4 | Model Score: 0.73
# Neighbors: 5 | Model Score: 0.73
# Neighbors: 6 | Model Score: 0.74
# Neighbors: 7 | Model Score: 0.74
# Neighbors: 8 | Model Score: 0.74
# Neighbors: 9 | Model Score: 0.74
```

The for loop results showed that model score was highest with 6-9 neighbors & lowest with 3 neighbors.



Plotted model results as a scatter.

```
plt.scatter(n_neighbors, scores, color="dodgerblue")
plt.title("KNN Model Score by # Neighbors")
plt.xlabel("# Neighbors")
plt.ylabel("Model Score")
```

KNN (K-Nearest Neighbor) vs. Logistic Regression

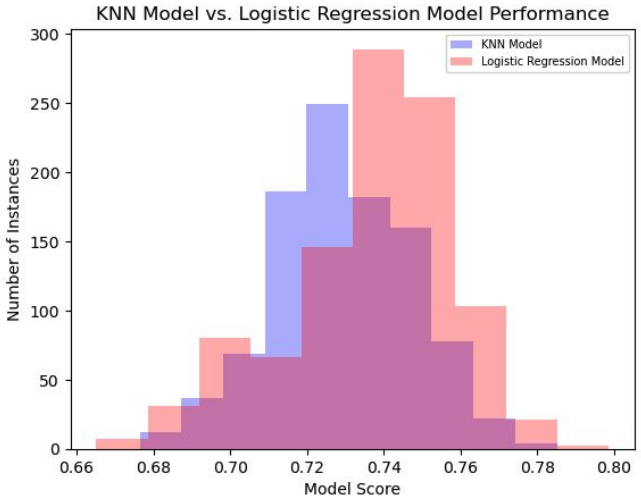
Ran for loops for both KNN & Logistic Regression models using the same data as prior.

```
scores_knn = []
for i in range(999):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    model_knn = KNeighborsClassifier(n_neighbors=5)
    model_knn.fit(X_train, y_train)
    scores_knn.append(model_knn.score(X_test,y_test))
```

```
scores_logistic = []
for i in range(999):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    model_logistic = linear_model.LogisticRegression()
    model_logistic.fit(X_train,y_train)
    scores_logistic.append(model_logistic.score(X_test,y_test))
```

Plotted results as a histogram to visualize & compare performances of the two models over 1,000 iterations.

```
plt.hist(scores_knn, color="blue", alpha=.3)
plt.hist(scores_logistic, color="red", alpha=.3)

plt.title("KNN Model vs. Logistic Regression Model Performance")
plt.xlabel("Model Score")
plt.ylabel("Number of Instances")

plt.legend(["KNN Model", "Logistic Regression Model"], ncol=1, loc="upper right", fontsize=7)
```

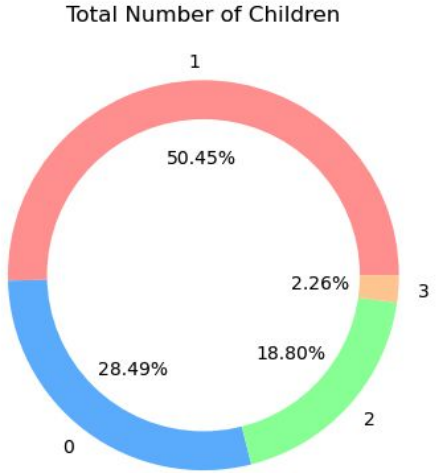


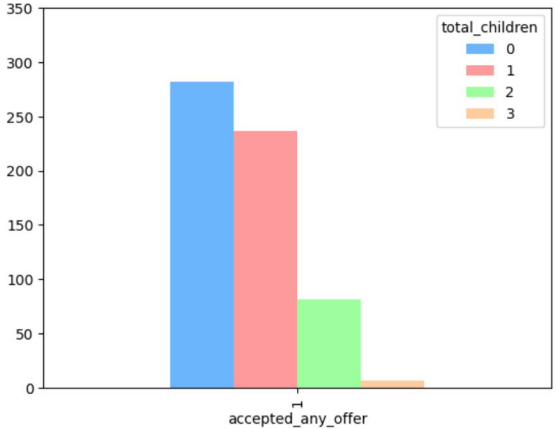
The histogram shows that over 1,000 iterations, the Logistic Regression model yielded more instances of stronger performance than the KNN model.

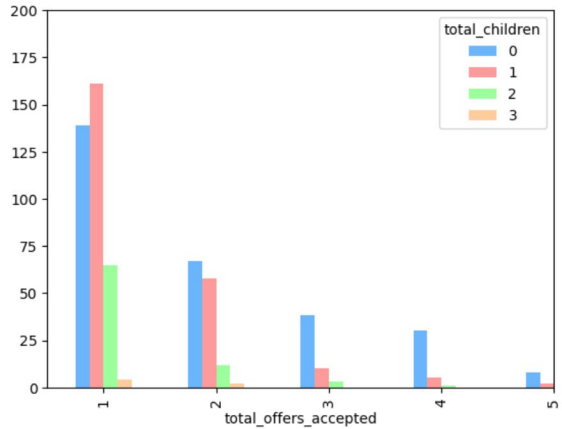
Data Exploration

Data Exploration

Created new column ‘total_children’ & plotted as a pie, then plotted relationships between total number of children & accepting any offer, as well as total number of children & total number of offers accepted as bars. Removed data where no offer was accepted to better visualize details of offer acceptances.





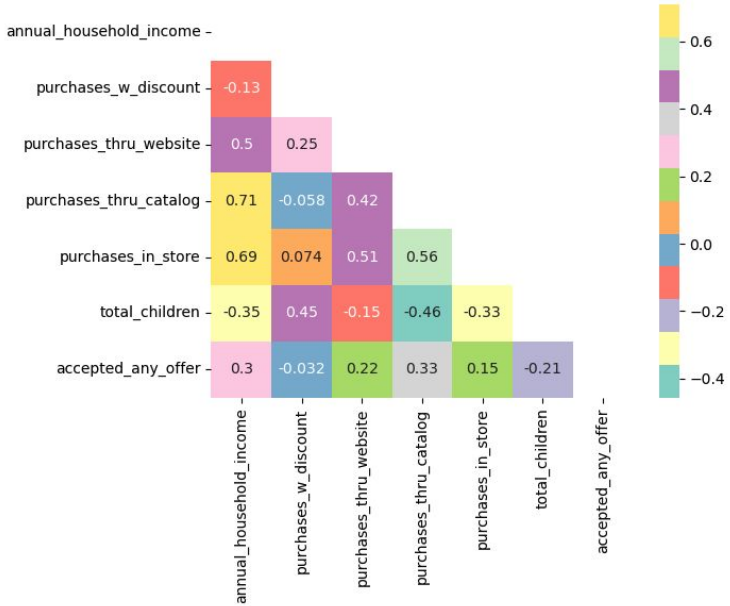



```
df["total_children"] = df["num_kids"] + df["num_teens"]

children = df["total_children"].value_counts()
values = children.values
labels = children.index
colors = ["#ff9999","#66b3ff","#99ff99","#ffcc99"]

plt.pie(values, labels=labels, colors = colors, autopct='%.2f%%')
plt.title("Total Number of Children")

center_circle = plt.Circle((0,0),0.80,fc='white')
fig = plt.gcf()
fig.gca().add_artist(center_circle);
```

```
accepted_any_offer_children = pd.crosstab(df["accepted_any_offer"], df["total_children"])
accepted_any_offer_children.plot(kind="bar", color=["#66b3ff","#ff9999","#99ff99","#ffcc99"])
plt.xlim([0.5,1.5])
plt.ylim([0,350]);
```

```
total_offers_accepted_children = pd.crosstab(df["total_offers_accepted"], df["total_children"])
total_offers_accepted_children.plot(kind="bar", color=["#66b3ff","#ff9999","#99ff99","#ffcc99"])
plt.xlim([0.5,5])
plt.ylim([0,200]);
```

Data Exploration Heatmap



Created new DataFrame ‘df_corr’ based off columns from df to view correlations between various forms of customer purchase as well as annual household income, total number of children, & offer acceptance.

```
df_corr = pd.DataFrame()

df_corr = df[["annual_household_income",
              "purchases_w_discount",
              "purchases_thru_website",
              "purchases_thru_catalog",
              "purchases_in_store",
              "total_children",
              "accepted_any_offer"]]
```

Built a heatmap to visualize strength of correlations, then cleaned heatmap to only show bottom triangle by creating a mask for upper triangle indices via numpy array.

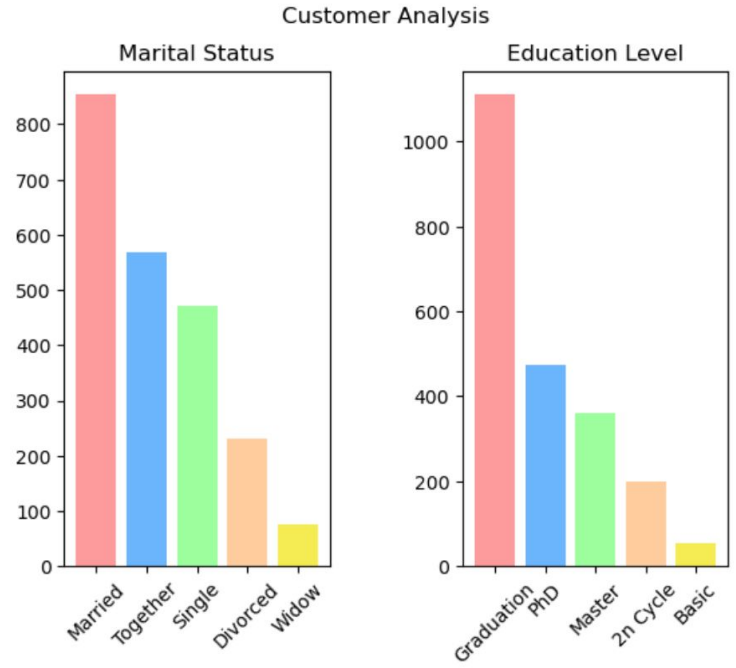
```
corr = df_corr.corr()
mask = np.zeros_like(corr)
triangle_indices = np.triu_indices_from(mask)
mask[triangle_indices] = True
sns.heatmap(corr, annot=True, mask=mask, cmap="Set3")
```

The heatmap shows strong correlations between annual household income & purchases through catalog (0.71) as well as purchases in store (0.69). It shows a relative correlation between purchases through catalog & purchases in store (0.56). Slight correlations are present between annual household income & purchases through website (0.5) as well as between total children & purchases with discount (0.45).

Data Exploration Subplot

Created a subplot to visualize customer distribution of marital status & education level.

```
marital = df["marital_status"].value_counts()
marital_values = marital.values
marital_labels = marital.index

education = df["education_level"].value_counts()
education_values = education.values
education_labels = education.index

fig, ax = plt.subplots(nrows=1, ncols=2)

ax[0].bar(marital_labels, marital_values, color=["#ff9999","#66b3ff","#99ff99","#ffcc99","#f5ec42"])
ax[0].set_title("Marital Status")
ax[0].set_xticklabels(labels=marital_labels, rotation=45)

ax[1].bar(education_labels, education_values, color=["#ff9999","#66b3ff","#99ff99","#ffcc99","#f5ec42"])
ax[1].set_title("Education Level")
ax[1].set_xticklabels(labels=education_labels, rotation=45)

plt.subplots_adjust(wspace = 0.5)
fig.suptitle("Customer Analysis");
```



Which model(s) leveraging marital status, education level, & annual household income data can most reliably predict whether a customer would accept an offer?

Multiple Models Logistic, KNN, Decision Tree, Random Forest, XGB

Initiated LabelEncoder to convert marital status & education levels to numeric values.

```
le_marital = LabelEncoder()
df["marital_status_le"] = le_marital.fit_transform(df["marital_status"])

le_education = LabelEncoder()
df["education_level_le"] = le_education.fit_transform(df["education_level"])
```

Defined IVs (X; numeric marital status & education level, annual_household_income) to train models to predict DV (y; accepted_any_offer). Introduced StandardScaler to normalize data & reduce biases in models.

```
X = df[["marital_status_le", "education_level_le", "annual_household_income"]]
y = df["accepted_any_offer"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Created empty lists ‘models_list’ & ‘score_list’ & trained Logistic Regression, KNN, Decision Tree, Random Forest, & XGBoost models for comparison. Appended model names & scores to respective lists for visual comparison via scatter.

```
models_list = []
score_list = []

models = (("LogisticRegression", LogisticRegression()),
          ("KNeighborsClassifier", KNeighborsClassifier()),
          ("DecisionTreeClassifier", DecisionTreeClassifier()),
          ("RandomForestClassifier", RandomForestClassifier()),
          ("XGBClassifier", XGBClassifier()))

for model_name, model in models:
    model.fit(X_train, y_train)
    models_list.append(model_name)
    score_list.append(round(model.score(X_test, y_test), 4))
```

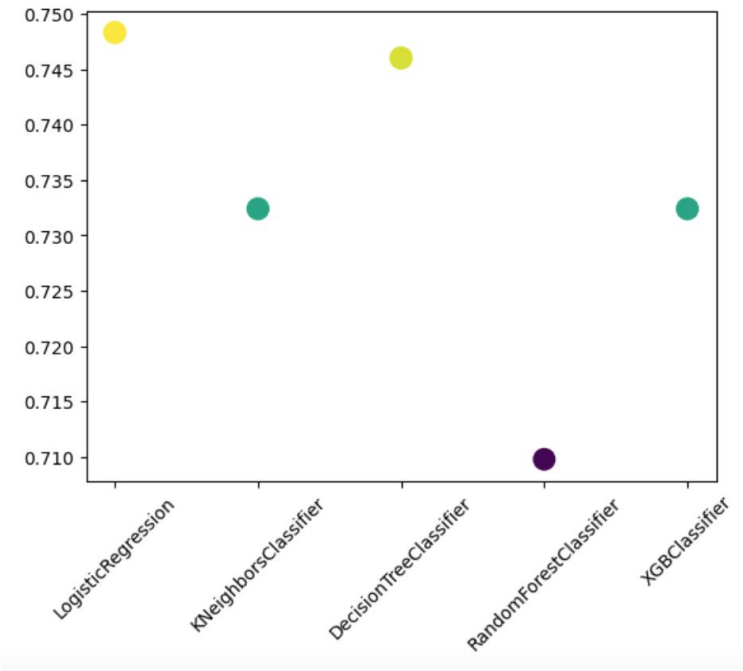
Multiple Models Logistic, KNN, Decision Tree, Random Forest, XGB

Plotted models as a scatter to visualize performance by model for the data used.

```
plt.scatter(models_list,
            score_list,
            s=[i*200 for i in score_list],
            c=score_list)
plt.xticks(rotation=45)
```

Of the five machine learning models tested, the Logistic Regression showed the strongest results, followed by Decision Tree. KNN & XGB performed similarly in the middle, while Random Forest returned the worst outcomes in predicting offer acceptance.

While these were the reflected results, a range of scores between 70-75% suggest that the models still pose notable room for error. As such, additional scrutiny into refining X & y variables should be considered when leveraging this data to inform decision-making processes.



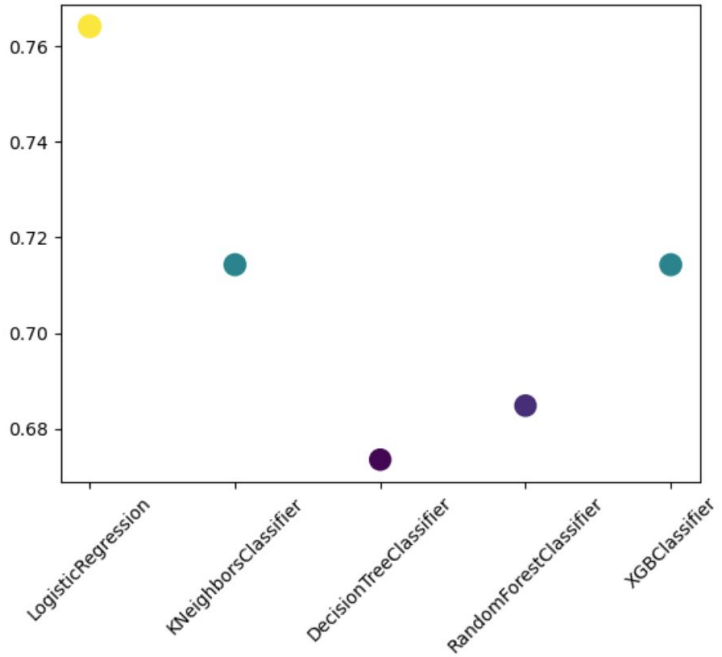
Multiple Models Logistic, KNN, Decision Tree, Random Forest, XGB

Repeated the same process using MinMaxScaler in place of StandardScaler.

```
X = df[["marital_status_le", "education_level_le", "annual_household_income"]]
y = df["accepted_any_offer"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
sc = MinMaxScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

When MinMaxScaler is used as a normalizer in place of StandardScaler, Logistic Regression remains the most promising model. KNN & XGB remain similar while Decision Tree model performance falls to the bottom below Random Forest.



KMeans

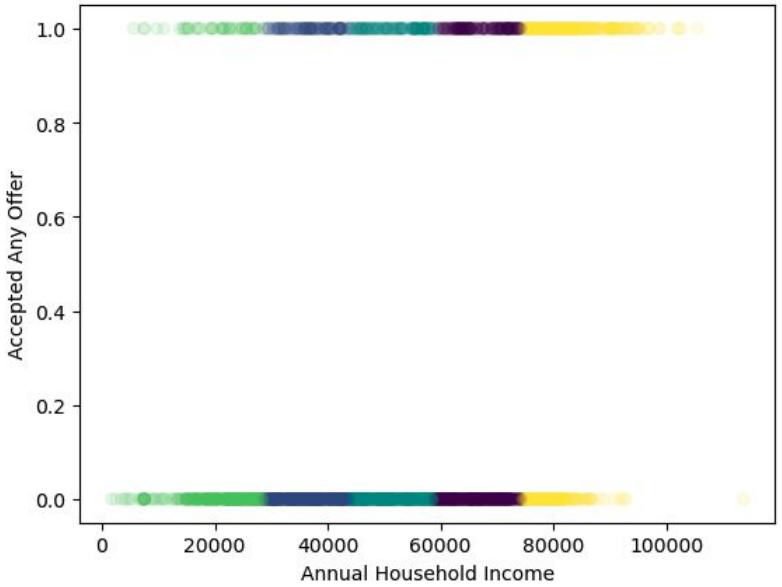
Used the same IVs (X) to fit a KMeans model into five clusters. Plotted annual_household_income against accepted_any_offer to visualize dataset groupings by outcomes.

```
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X)
```

```
/Users/npcheng/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:870: FutureWa
e of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
  warnings.warn(
```

```
KMeans(n_clusters=5, random_state=42)
```

```
labels = kmeans.labels_
plt.scatter(df["annual_household_income"], df["accepted_any_offer"], c=labels, alpha=0.1)
```



The scatter helps to visualize groups within the data, While there is notable overlap between annual household incomes of 40,000 & 80,000, the results of customers with annual household incomes <40,000 were more greatly concentrated in 0 (did not accept any offer), whereas those with annual household incomes >80,000 were more greatly concentrated in 1 (accepted any offer).

The Findings

A broad range of machine learning models were trained & tested against various data points (IVs & DVs) across the marketing dataset. While the majority yielded results below 80% accuracy, these tests were a promising start in understanding how to consider various inputs & their relationships, correlations, as well as capacities to predict one another.

Across the board, annual household income was a strong IV to base models off, however companies can also glean insights from the nuances of spend on different products, marital status, education level, number of children, & different avenues of purchase. These, as well as additional data points leveraged, can be employed to better understand consumer behavior, internal & external influences, & improve companies' segmentation capabilities.

---

https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign?select=marketing_campaign.csv