Data_100_Final_-_Notebook

June 5, 2020

1 INVESTIGATING CONTRACEPTIVES

1.1 *Sara Imam, Shirley Wang, Sierra Dean*

1.1.1 *DATA 100*

The use (or lack thereof) of contraceptive methods can be strongly affected by demographic and socio-economic factors. Being able to predict the use from the factors, allows for better marketing or advertising to targeted groups. With this goal, we attempted to be able to predict the use of contraceptives (no use, short term, long term) from looking at certain demographic and socio-economic features.

2 A. Imports and Data Cleaning

[44]:
```
from sklearn.feature_extraction import DictVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn import ensemble
import plotly.express as px
from sklearn import tree
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

[45]:
```
#importing Data
contraceptives = pd.read_csv("contraceptive_for_students.csv")
```

```
contraceptives.head()
```

[45]:
```
   wife_age  wife_education  husband_education  num_child  wife_religion  \
0        24               2                  3          3              1
1        45               1                  3         10              1
2        43               2                  3          7              1
3        42               3                  2          9              1
4        36               3                  3          8              1

   wife_work  husband_occupation  standard_living  media_exposure  \
0          1                   2                3               0
1          1                   3                4               0
2          1                   3                4               0
3          1                   3                3               0
4          1                   3                2               0

   contraceptive
0              1
1              1
2              1
3              1
4              1
```

[46]:
```
#cleaning data with one hot encoding
import warnings; warnings.simplefilter('ignore')

oh_enc = OneHotEncoder()
oh_enc.fit(contraceptives[['wife_education', 'husband_education',␣
↪'husband_occupation', 'standard_living']])
ohe_transformed = oh_enc.transform(contraceptives[['wife_education',␣
↪'husband_education', 'husband_occupation', 'standard_living']]).todense()
contraceptives_ohe = pd.DataFrame(ohe_transformed)
contraceptives_ohe = contraceptives_ohe.rename({0:'wife education: 1', 1: 'wife␣
↪education: 2', 2: 'wife education: 3', 3: 'wife education: 4',
                                                4: 'husband education: 1', 5:␣
↪'husband education: 2', 6: 'husband education: 3', 7: 'husband education: 4',
                                               8: 'husband occupation: 1', 9:␣
↪'husband occupation: 2', 10: 'husband occupation: 3', 11: 'husband␣
↪occupation: 4',
                                               12: 'standard_living: 1', 13:␣
↪'standard_living: 2', 14: 'standard_living: 3', 15: 'standard_living: 4'},␣
↪axis = 1)

contraceptives_cleaned = contraceptives_ohe.join(contraceptives).drop(columns =␣
↪{'wife_education', 'husband_education', 'husband_occupation',␣
↪'standard_living'})
```

```
contraceptives_cleaned.head()
```

[46]:

```
   wife education: 1  wife education: 2  wife education: 3  wife education: 4  \
0                0.0                1.0                0.0                0.0
1                1.0                0.0                0.0                0.0
2                0.0                1.0                0.0                0.0
3                0.0                0.0                1.0                0.0
4                0.0                0.0                1.0                0.0

   husband education: 1  husband education: 2  husband education: 3  \
0                   0.0                   0.0                   1.0
1                   0.0                   0.0                   1.0
2                   0.0                   0.0                   1.0
3                   0.0                   1.0                   0.0
4                   0.0                   0.0                   1.0

   husband education: 4  husband occupation: 1  husband occupation: 2  …  \
0                   0.0                    0.0                    1.0  …
1                   0.0                    0.0                    0.0  …
2                   0.0                    0.0                    0.0  …
3                   0.0                    0.0                    0.0  …
4                   0.0                    0.0                    0.0  …

   standard_living: 1  standard_living: 2  standard_living: 3  \
0                 0.0                 0.0                 1.0
1                 0.0                 0.0                 0.0
2                 0.0                 0.0                 0.0
3                 0.0                 0.0                 1.0
4                 0.0                 1.0                 0.0

   standard_living: 4  wife_age  num_child  wife_religion  wife_work  \
0                 0.0        24          3              1          1
1                 1.0        45         10              1          1
2                 1.0        43          7              1          1
3                 0.0        42          9              1          1
4                 0.0        36          8              1          1

   media_exposure  contraceptive
0               0              1
1               0              1
2               0              1
3               0              1
4               0              1

[5 rows x 22 columns]
```

[47]:
```
#Splitting data into Training and Testing Data for Contraceptive Classifier
X = contraceptives_cleaned.loc[:, "wife education: 1":"media_exposure"]
Y = contraceptives_cleaned.loc[:, 'contraceptive']

#80:20 train test split, Seeding for consistency across groupmates
np.random.seed(41)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20)
```

3 B. Visualizations

The following visualizations were made for the following categories: 1. Wife's age (numerical) 2. Wife's education (categorical) 1=low, 2, 3, 4=high 3. Husband's education (categorical) 1=low, 2, 3, 4=high 4. Number of children ever born (numerical) 5. Wife's religion (binary) 0=Non-Islam, 1=Islam 6. Wife's now working? (binary) 0=Yes, 1=No 7. Husband's occupation (categorical) 1, 2, 3, 4 8. Standard-of-living index (categorical) 1=low, 2, 3, 4=high 9. Media exposure (binary) 0=Good, 1=Not good

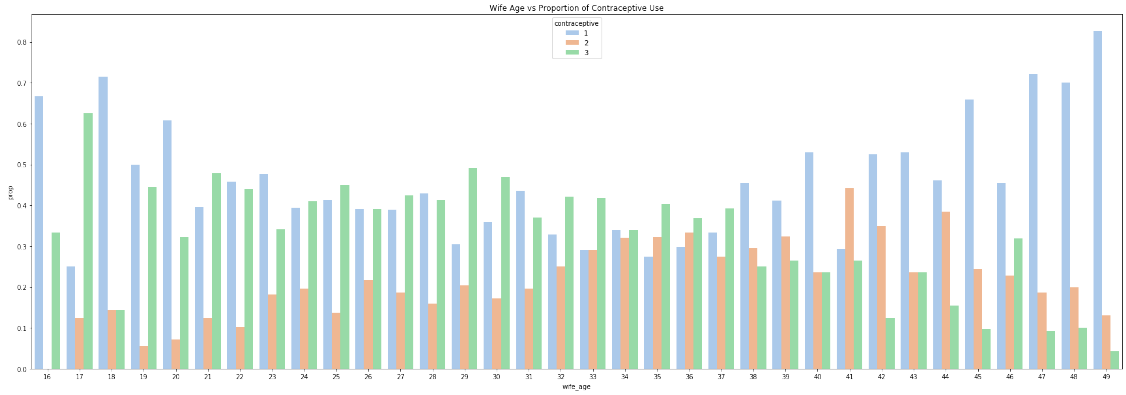
The goal of each visualization was to find a correlation between (1-9) and either Number of children, Contraception used, or both. Some of the data was normalized to show percentages.

[48]:
```
# set color for all the following visualizations:
sns.set_palette("pastel")
```

3.0.1 1. Wife Age

[49]:
```
# wife age vs proportion of contraception used for each age group
proportion_wifeeage = contraceptives["contraceptive"].
↪groupby(contraceptives["wife_age"]).value_counts(normalize=True).
↪rename("prop").reset_index()
plt.figure(figsize=(30,10))
sns.barplot(x="wife_age", y="prop", hue="contraceptive",␣
↪data=proportion_wifeeage)
plt.title("Wife Age vs Proportion of Contraceptive Use")
```

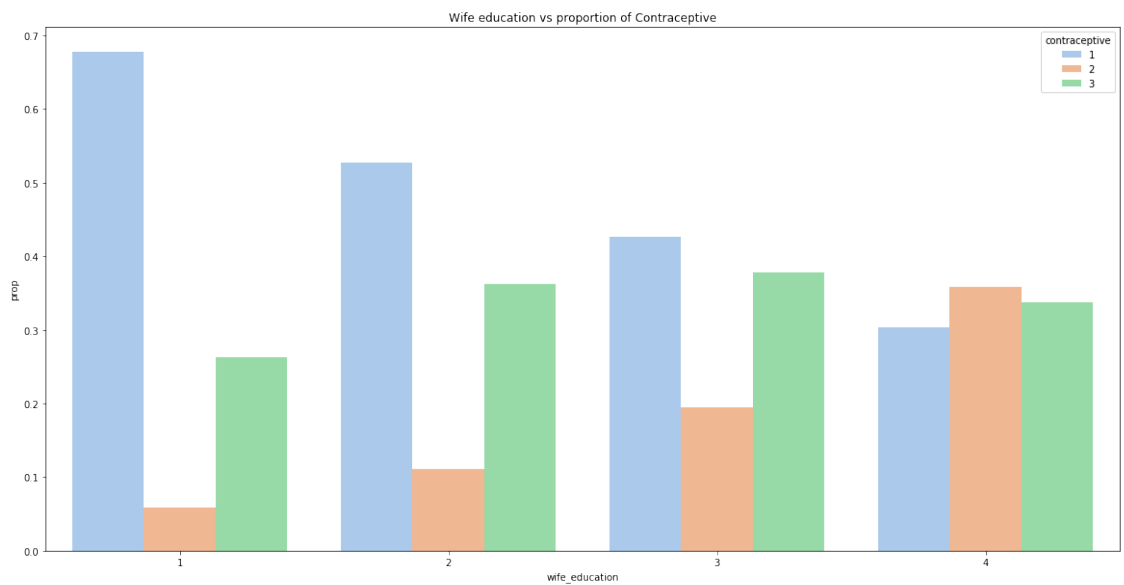
[49]:
```
Text(0.5, 1.0, 'Wife Age vs Proportion of Contraceptive Use')
```



3.0.2 2. Wife Education

[50]:
```
# wife education vs prop of contraceptive
proportion_wifeeducation = contraceptives["contraceptive"].
 ↪groupby(contraceptives["wife_education"]).value_counts(normalize=True).
 ↪rename("prop").reset_index()
plt.figure(figsize=(20,10))
sns.barplot(x="wife_education", y="prop", hue="contraceptive",␣
 ↪data=proportion_wifeeducation)
plt.title("Wife education vs proportion of Contraceptive")
```

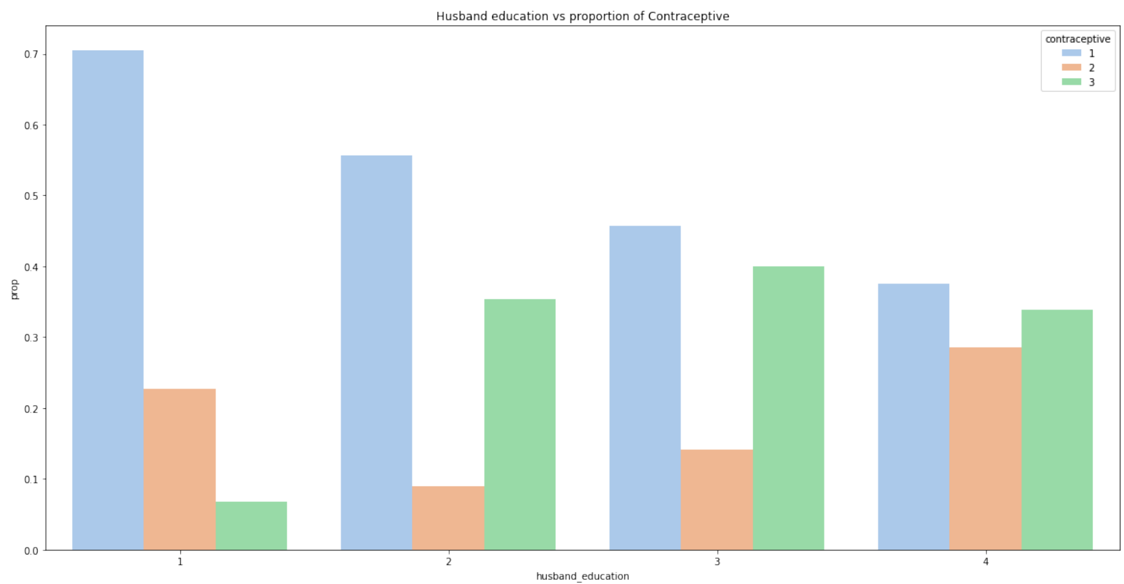
[50]:
```
Text(0.5, 1.0, 'Wife education vs proportion of Contraceptive')
```



3.0.3 3. Husband Education

[51]:
```
# husband education vs prop of contraceptive
proportion_husbandeducation = contraceptives["contraceptive"].
↪groupby(contraceptives["husband_education"]).value_counts(normalize=True).
↪rename("prop").reset_index()
plt.figure(figsize=(20,10))
sns.barplot(x="husband_education", y="prop", hue="contraceptive",␣
↪data=proportion_husbandeducation)
plt.title("Husband education vs proportion of Contraceptive")
```

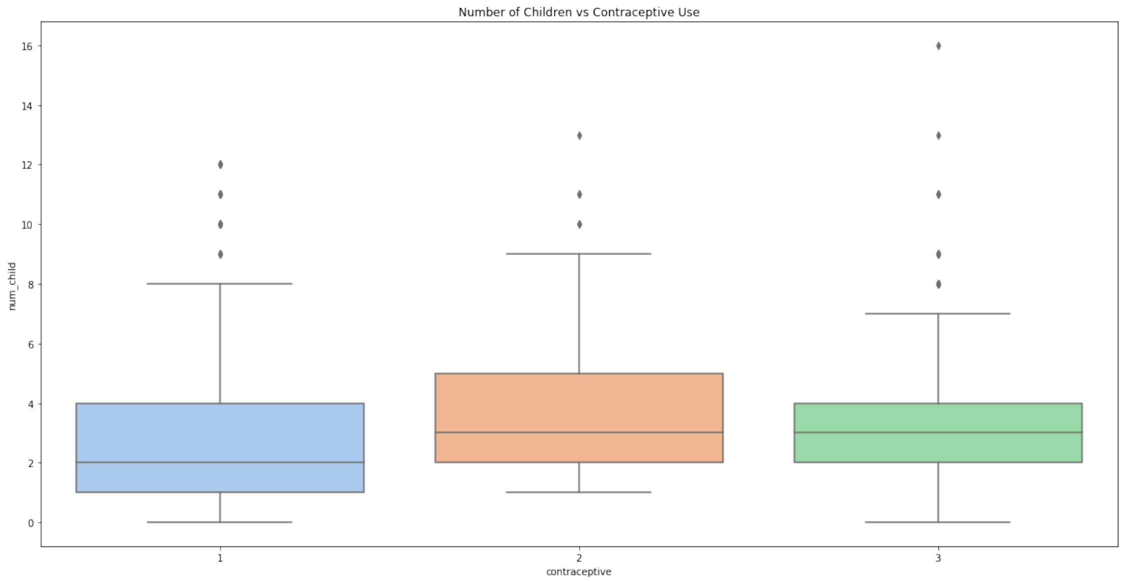
[51]: Text(0.5, 1.0, 'Husband education vs proportion of Contraceptive')



3.0.4 4. Number of Children Ever Born

[52]:
```
# Number of children vs prop of contraceptives
num_children_contraceptive = contraceptives[['num_child', 'contraceptive']]
plt.figure(figsize=(20,10))
sns.boxplot(x = "contraceptive", y = "num_child",␣
↪data=num_children_contraceptive)
plt.title("Number of Children vs Contraceptive Use")
```

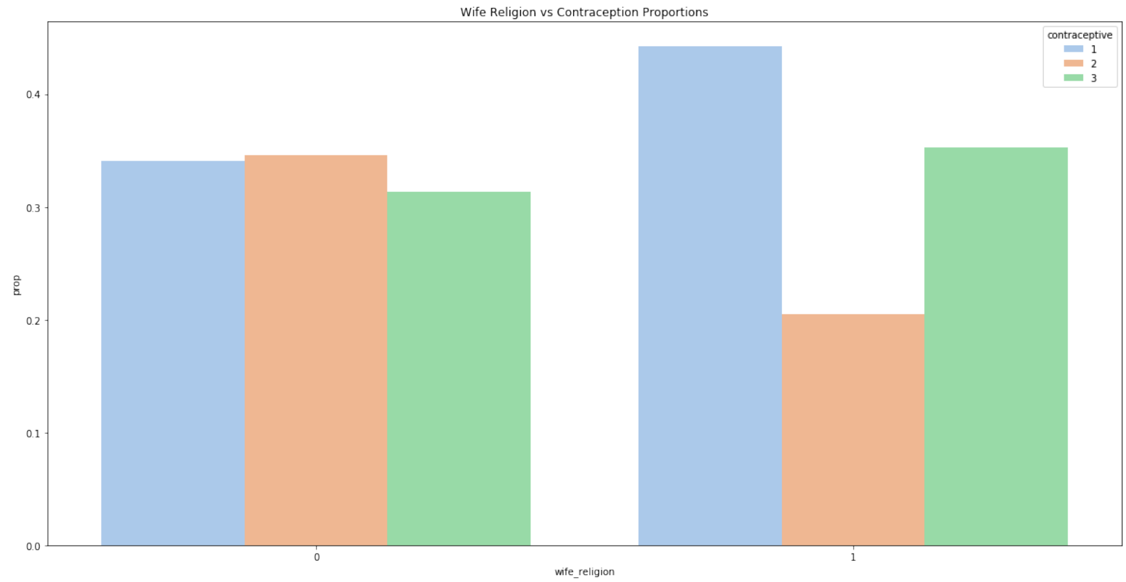
[52]: Text(0.5, 1.0, 'Number of Children vs Contraceptive Use')



3.0.5 5. Wife's Religion

[53]:
```
# wife religion vs prop of contraceptive
cleaned = contraceptives["contraceptive"].
 ↪groupby(contraceptives["wife_religion"]).value_counts(normalize=True).
 ↪rename("prop").reset_index()
plt.figure(figsize=(20,10))
x = sns.barplot(x="wife_religion", y="prop", hue="contraceptive", data=cleaned)
x.set_title("Wife Religion vs Contraception Proportions")
```

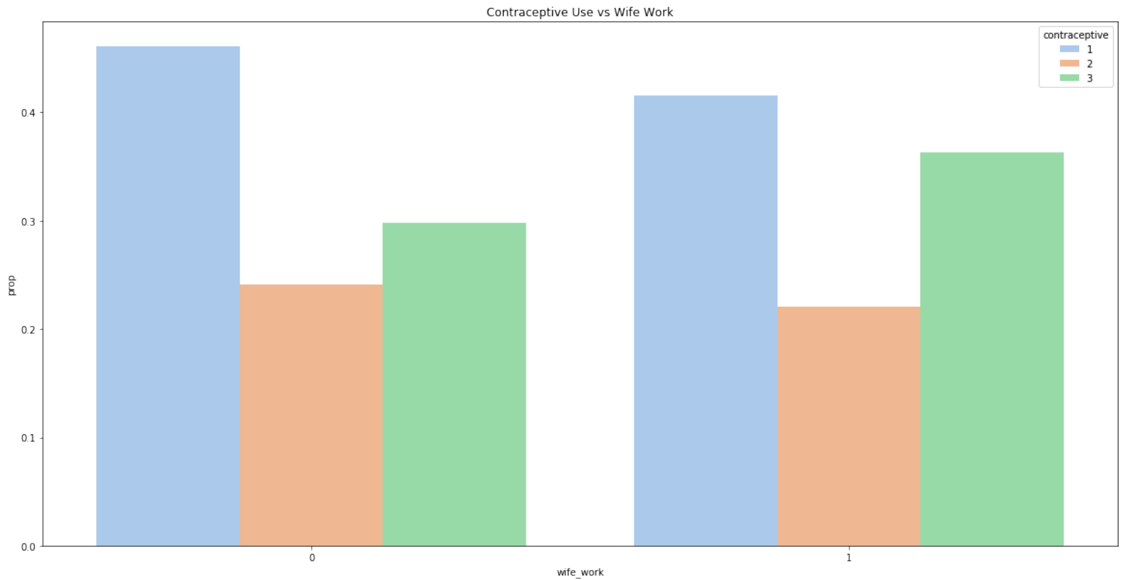
[53]:
```
Text(0.5, 1.0, 'Wife Religion vs Contraception Proportions')
```



3.0.6 6. Wife now working?

[54]:
```
# Contraceptive use vs wifework
proportion_wifework= contraceptives["contraceptive"].
↪groupby(contraceptives["wife_work"]).value_counts(normalize=True).
↪rename("prop").reset_index()
plt.figure(figsize=(20,10))
x = sns.barplot(x="wife_work", y="prop", hue="contraceptive",␣
↪data=proportion_wifework)
x.set_title("Contraceptive Use vs Wife Work")
```

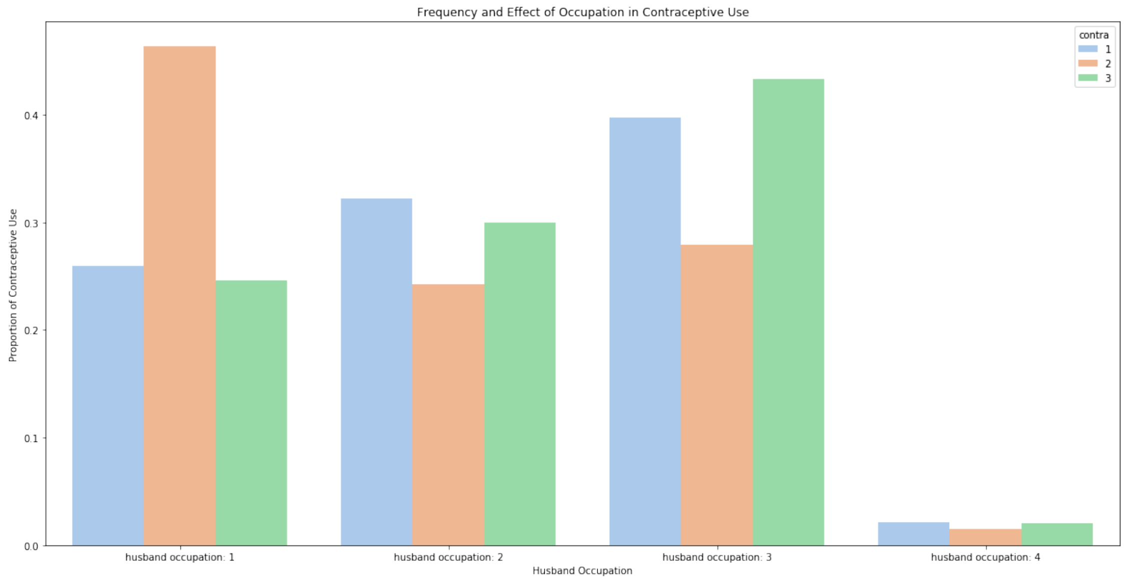
[54]:
```
Text(0.5, 1.0, 'Contraceptive Use vs Wife Work')
```



3.0.7 7. Husbands Occupation

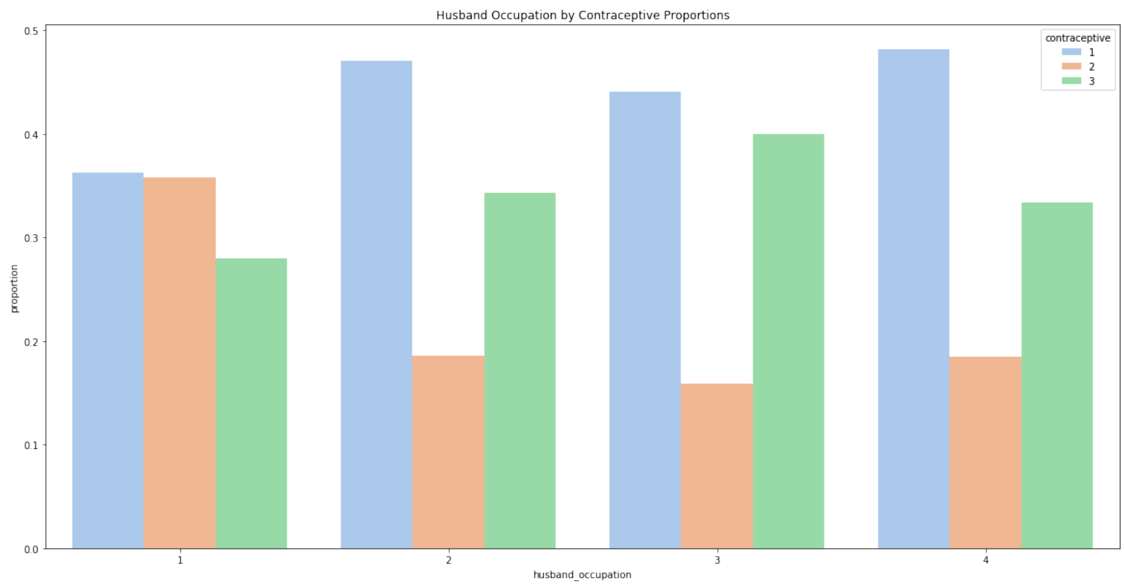
[55]:
```python
# a. Frequency and Effect of Occupation in Contraceptive Use
import warnings; warnings.simplefilter('ignore')

barplt = X_train[["husband occupation: 1", "husband occupation: 2", "husband␣
↪occupation: 3", "husband occupation: 4"]]

barplt["contra"] = Y_train
barplt = barplt.melt("contra")
grouped = barplt.groupby(["variable","contra"]).mean()

grouped = grouped.reset_index()
plt.figure(figsize=(20,10))
x = sns.barplot(x="variable", y="value",data=grouped, hue="contra")
x.set_title("Frequency and Effect of Occupation in Contraceptive Use")
x.set(xlabel="Husband Occupation", ylabel="Proportion of Contraceptive Use")
```

[55]:
```
[Text(0, 0.5, 'Proportion of Contraceptive Use'),
 Text(0.5, 0, 'Husband Occupation')]
```



[56]:
```python
# b. Husband Occupation by Contraceptive Proportions
proportion_wifeeducation = contraceptives["contraceptive"].
  ↪groupby(contraceptives["husband_occupation"]).value_counts(normalize=True).
  ↪rename("proportion").reset_index()
plt.figure(figsize=(20,10))
x = sns.barplot(x="husband_occupation", y="proportion", hue="contraceptive",␣
  ↪data=proportion_wifeeducation)
x.set_title("Husband Occupation by Contraceptive Proportions")
```

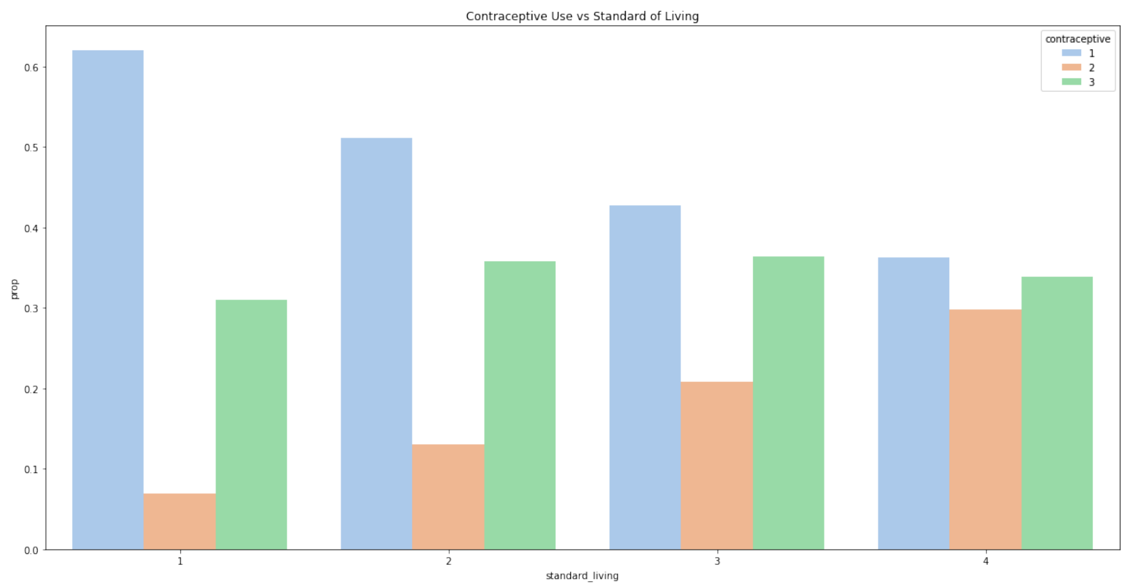
[56]: Text(0.5, 1.0, 'Husband Occupation by Contraceptive Proportions')



3.0.8 8. Standard of Living Index

[57]:
```
# standard of living cs prop of contraceptive
proportion_standardliving = contraceptives["contraceptive"].
↪groupby(contraceptives["standard_living"]).value_counts(normalize=True).
↪rename("prop").reset_index()
plt.figure(figsize=(20,10))
x = sns.barplot(x="standard_living", y="prop", hue="contraceptive",␣
↪data=proportion_standardliving)
x.set_title("Contraceptive Use vs Standard of Living")
```

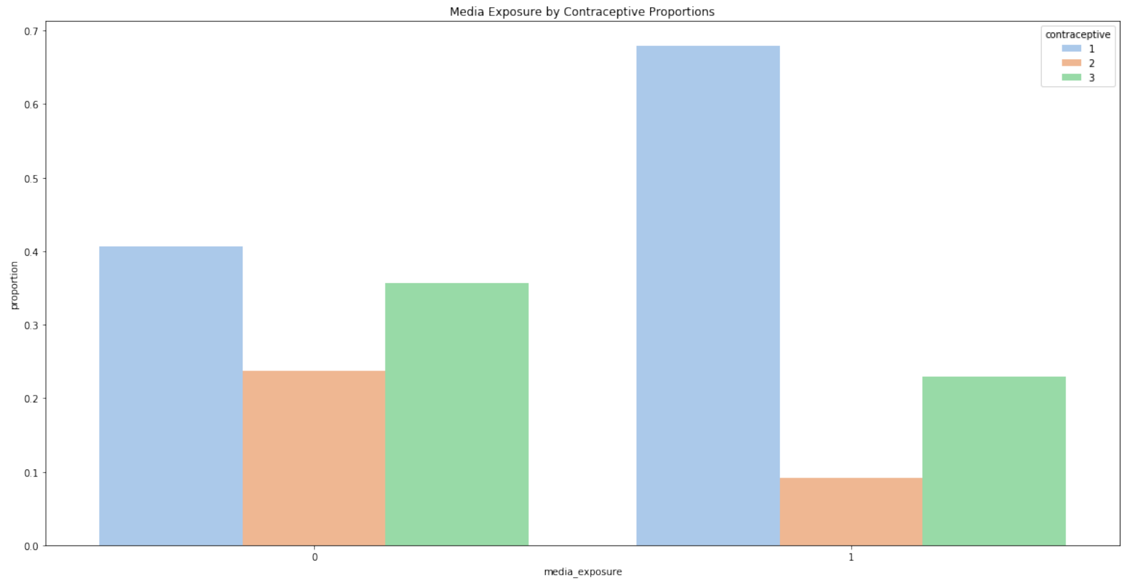
[57]:
```
Text(0.5, 1.0, 'Contraceptive Use vs Standard of Living')
```



3.0.9 9. Media exposure

[58]:
```
proportion_media = contraceptives["contraceptive"].
↪groupby(contraceptives["media_exposure"]).value_counts(normalize=True).
↪rename("proportion").reset_index()
plt.figure(figsize=(20,10))
x = sns.barplot(x="media_exposure", y="proportion", hue="contraceptive",␣
↪data=proportion_media)
x.set_title("Media Exposure by Contraceptive Proportions")
```

[58]: Text(0.5, 1.0, 'Media Exposure by Contraceptive Proportions')



4 D. Model

The following is all models used, outputting the training accuracy for each model. We outputted testing accuracy only for best models. We selected our initial Features based on the visualizations.

4.0.1 1. Contraception Classifier

[59]:
```
# Logistic Regression Model with Cross Validation
X_features = X_train[["wife_work", "num_child", "standard_living: 1", 
 ↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",
 ↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:
 ↪3", "wife education: 4", "husband education: 1", "husband education: 2",
 ↪"husband education: 3", "husband education: 4"]]

model_cv_lr = LogisticRegressionCV(cv=5, random_state=42, multi_class="ovr").
 ↪fit(X_features, Y_train)
model_cv_lr.predict(X_features)
training_acc = model_cv_lr.score(X_features, Y_train)
print("Training Accuracy:", training_acc)
```

Training Accuracy: 0.4855687606112054

[60]:
```
#Decision Tree Model
X_features = X_train[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]
X_features_test = X_test[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]

model_dt = tree.DecisionTreeClassifier()
model_dt.fit(X_features, Y_train)
training_acc = model_dt.score(X_features, Y_train)
testing_acc = model_dt.score(X_features_test, Y_test)
print("Training Accuracy:", training_acc, "Testing Accuracy:", testing_acc)
```

```
Training Accuracy: 0.7156196943972836 Testing Accuracy: 0.44745762711864406
```

[61]:
```
#Random Forest Model
X_features = X_train[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]
X_features_test = X_test[["wife_work","num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]

model_rf = ensemble.RandomForestClassifier(random_state = 42, n_estimators = 20)
model_rf.fit(X_features, Y_train)
training_acc = model_rf.score(X_features, Y_train)
testing_acc = model_rf.score(X_features_test, Y_test)
print("Training Accuracy:", training_acc, "Testing Accuracy:", testing_acc)
```

```
Training Accuracy: 0.7147707979626485 Testing Accuracy: 0.4576271186440678
```

Trying new Features After determining that the random forest was the best classifier, we attempted to make it better using different features.

[62]:
```
#Random Forest Model Trying Everything
X_features = X_train
```

```
model_rf_newX = ensemble.RandomForestClassifier(random_state = 42, n_estimators␣
↪= 20)
model_rf_newX.fit(X_features, Y_train)
training_acc = model_rf_newX.score(X_features, Y_train)
#add testing back
print("Training Accuracy:", training_acc)
```

```
Training Accuracy: 0.9541595925297114
```

[63]:
```
#Random Forest Model with removed num_child and added wife_religion, media␣
↪exposure
X_features = X_train[["wife_work", "standard_living: 1",  "standard_living: 2",␣
↪ "standard_living: 3",  "standard_living: 4", "media_exposure", "wife␣
↪education: 1", "wife education: 2", "wife education: 3", "wife education:␣
↪4", "husband education: 1", "husband education: 2", "husband education: 3",␣
↪"husband education: 4", "media_exposure", "wife_religion"]]

model_rf_newX2 = ensemble.RandomForestClassifier(random_state = 42,␣
↪n_estimators = 20)
model_rf_newX2.fit(X_features, Y_train)
training_acc = model_rf_newX2.score(X_features, Y_train)
print("Training Accuracy:", training_acc)
```

```
Training Accuracy: 0.5466893039049237
```

[64]:
```
#Random Forest Model without standard living and anything to do with husband
X_features = X_train[["wife_work", "wife_religion", "num_child",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4"]]

model_rf_newX3 = ensemble.RandomForestClassifier(random_state = 42,␣
↪n_estimators = 20)
model_rf_newX3.fit(X_features, Y_train)
training_acc = model_rf_newX3.score(X_features, Y_train)
print("Training Accuracy:", training_acc)
```

```
Training Accuracy: 0.5933786078098472
```

[65]:
```
#Random Forest Model with same as above, but with husband education.
X_features = X_train[["wife_work", "wife_religion", "num_child",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", 'husband education: 1', 'husband education:␣
↪2','husband education: 3', 'husband education: 4']]
X_features_test = X_test[["wife_work", "wife_religion", "num_child",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", 'husband education: 1', 'husband education:␣
↪2','husband education: 3', 'husband education: 4']]
```

```
model_rf_newX4 = ensemble.RandomForestClassifier(random_state = 42,␣
 ↪n_estimators = 20)
model_rf_newX4.fit(X_features, Y_train)
training_acc = model_rf_newX4.score(X_features, Y_train)
testing_acc = model_rf_newX4.score(X_features_test, Y_test)
print("Training Accuracy:", training_acc, "Testing Accuracy:", testing_acc)
```

```
Training Accuracy: 0.6528013582342954 Testing Accuracy: 0.4711864406779661
```

Trying Hyperparameters After determining that our features were good (since none of the other features improved it), we decided to try tuning hyperparameters to improve our model.

[66]:
```
#Random forest model with max_features limited. Best model yet!
X_features = X_train[["wife_work", "num_child", "standard_living: 1", ␣
 ↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
 ↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
 ↪3", "wife education: 4", "husband education: 1","husband education: 2",␣
 ↪"husband education: 3", "husband education: 4"]]
X_test_features = X_test[["wife_work", "num_child", "standard_living: 1", ␣
 ↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
 ↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
 ↪3", "wife education: 4", "husband education: 1","husband education: 2",␣
 ↪"husband education: 3", "husband education: 4"]]

model_rf_hp = ensemble.RandomForestClassifier(random_state = 42, n_estimators =␣
 ↪20, max_features=10)
model_rf_hp.fit(X_features, Y_train)

training_acc = model_rf_hp.score(X_features, Y_train)
test_acc = model_rf_hp.score(X_test_features, Y_test)
print("Training Accuracy:", training_acc, "Testing Accuracy:", testing_acc)
```

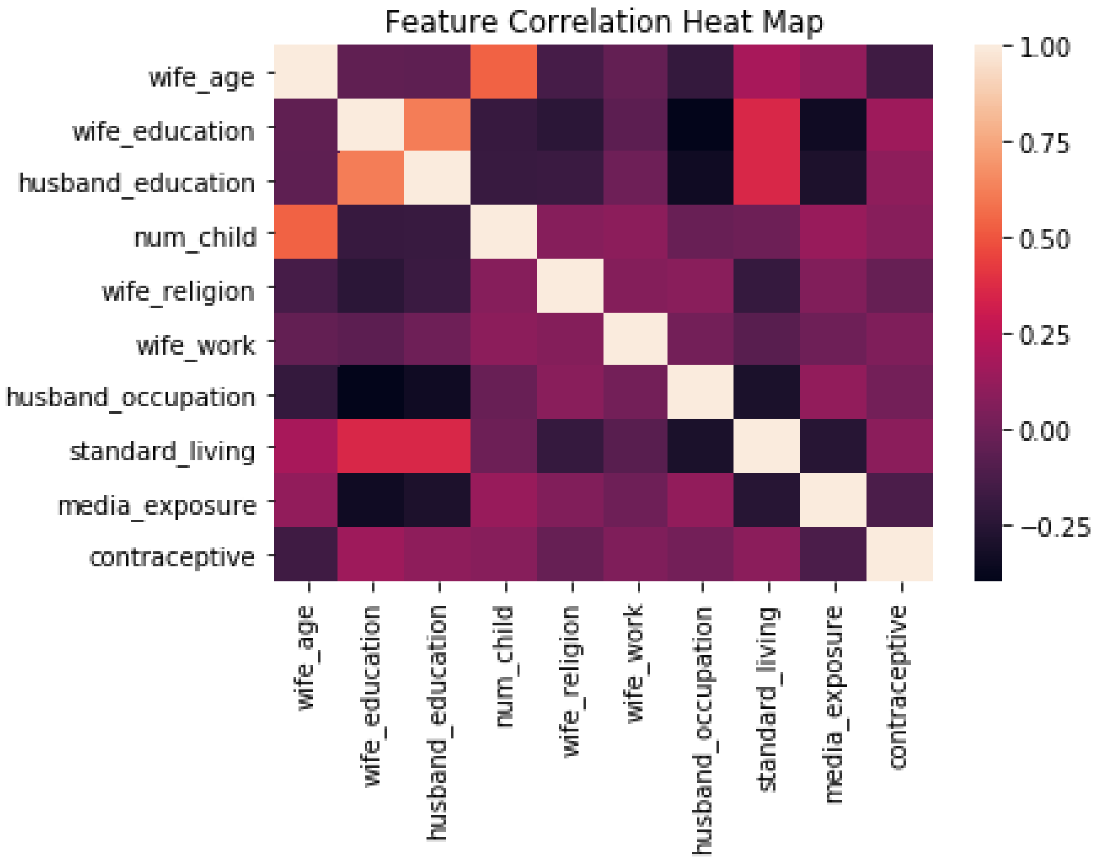
```
Training Accuracy: 0.7156196943972836 Testing Accuracy: 0.4711864406779661
```

5 E. Analyzing Models

5.0.1 1. Overall data correlation

[67]:
```
# Feature Correlation Heat Map
plt = sns.heatmap(contraceptives.corr())
plt.set_title("Feature Correlation Heat Map")
```

[67]: `Text(0.5, 1, 'Feature Correlation Heat Map')`



5.0.2 2. Checking for Class Imbalance

[69]:
```
#class imbalance

class_1 = np.sum(Y == 1)
class_2 = np.sum(Y == 2)
class_3 = np.sum(Y == 3)
print("number of class 1:", class_1)
print("number of class 2:", class_2)
print("number of class 3:", class_3)
```

```
number of class 1: 629
number of class 2: 333
number of class 3: 511
```

5.0.3 3. Analyzing Confusion Matrix and Precision/Recall

[70]:
```
#confusion Matrix

#true values
labels = Y_train
#predicted
predictions =  model_rf_hp.predict(X_features)

cm = confusion_matrix(labels, predictions)
cm

# (y = TrueValues )
# 1
# 2
# 3
#   1  2  3 (x = predicted)
```

[70]:
```
array([[400,  49,  67],
       [ 46, 178,  48],
       [ 57,  68, 265]], dtype=int64)
```

[72]:
```
recall = np.diag(cm) / np.sum(cm, axis = 1)
precision = np.diag(cm) / np.sum(cm, axis = 0)

avgrecall = np.mean(recall)
avgprecision = np.mean(precision)
print("recall[1,2,3]:", recall, "precision[1,2,3]:", precision)
print("average recall:", avgrecall, " average precision:", avgprecision)
```

```
recall[1,2,3]: [0.7751938  0.65441176 0.67948718] precision[1,2,3]: [0.79522863
0.60338983 0.69736842]
average recall: 0.7030309142142247  average precision: 0.6986622932639075
```

6 Model: Second Attempt

After realizing our model did not perform well, we decided to try another apporach at classifying: a binary model

[73]:
```
#Binary Classifier: Making a new Y
new_Y = [1 if x == 2 or x == 3 else 0 for x in Y_train]
```

[74]:
```
#linear regression model
import warnings; warnings.simplefilter('ignore')
```

```
X_features = X_train[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]


model_linr = LogisticRegression().fit(X_features, new_Y)
Y_pred = model_linr.predict(X_features)
model_linr.score(X_features, new_Y)
```

[74]: 0.6536502546689303

[75]:
```
#decison tree model
new_Y_test = [1 if x == 2 or x == 3 else 0 for x in Y_test]
X_features = X_train[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]
X_features_test = X_test[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]


model_dt_binary = tree.DecisionTreeClassifier()
model_dt_binary.fit(X_features, new_Y)
training_acc = model_dt_binary.score(X_features, new_Y)
testing_acc = model_dt_binary.score(X_features_test, new_Y_test)
print("Training Accuracy:", training_acc, "Testing Accuracy:", testing_acc)
```

Training Accuracy: 0.8234295415959253 Testing Accuracy: 0.6745762711864407

[76]:
```
#random forest model
new_Y_test = [1 if x == 2 or x == 3 else 0 for x in Y_test]
X_features = X_train[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]
X_features_test = X_test[["wife_work", "num_child", "standard_living: 1", ␣
↪"standard_living: 2",  "standard_living: 3",  "standard_living: 4",␣
↪"media_exposure", "wife education: 1", "wife education: 2", "wife education:␣
↪3", "wife education: 4", "husband education: 1", "husband education: 2",␣
↪"husband education: 3", "husband education: 4"]]
```

```
model_rf = ensemble.RandomForestClassifier(random_state = 42, n_estimators =␣
↪20, max_features = 10)
model_rf.fit(X_features, new_Y)

training_acc = model_rf.score(X_features, new_Y)
testing_acc = model_rf.score(X_features_test, new_Y_test)
print("Training Accuracy:", training_acc, "Testing Accuracy:", testing_acc)
```

```
Training Accuracy: 0.8225806451612904 Testing Accuracy: 0.6677966101694915
```