# Binary Classification and Visualization of Yearly Salary Based on Demographics

Jeffrey Pan
1223505816
Arizona State University
Tempe, Arizona
Jpan59@my.asu.edu

## Introduction

As a data analyst at XYZ Corporation, I have been assigned a new project to work with UVW College in increasing enrollment through targeted marketing efforts. The college has identified salary as a key demographic and will be using data from the United States Census Bureau, with a focus on the $50,000 income threshold, to develop marketing profiles. The data will be analyzed to determine key variables such as age, gender, education status, marital status, and occupation to better understand the individuals making less than and more than $50,000. Then, create at least five user stories, paired with visualizations, that help paint a picture of ways that various demographics influence a student's salary.

## Goals

1. Develop marketing profiles using data supplied by the United States Census Bureau, with a focus on individuals earning more or less than $50,000 annually.
2. Identify the key variables that determine an individual's income and group them for use in developing an application to predict income.
3. Create an application that can predict an individual's income based on different input parameters, allowing the marketing team to tailor their efforts when reaching out to potential students.

## Business Objectives

1. Gain a solid understanding of the market demographics to better tailor marketing efforts.
2. Use the application to predict the income of potential students based on different input parameters to tailor marketing efforts and increase the likelihood of enrollment.

## Assumptions

Business Assumption #1 – UVW College wants to bolster its enrolled by developing targeting marketing efforts based on the market profiles created.

Business Assumption #2 – UVW College is using salary as a key demographic to determine the criteria for marketing its degree programs.

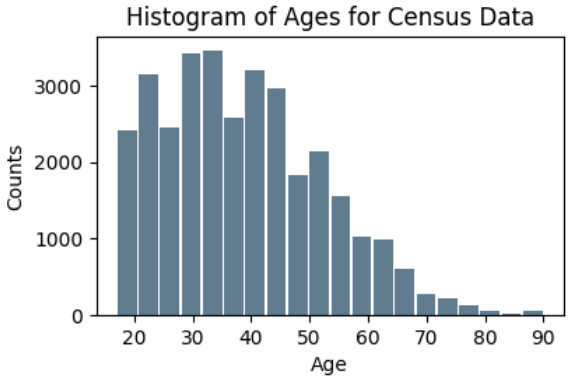
Technical Assumption #1 – The data source used to analyze the target market is based on a 1994 US Census database. This database has a possibility to be biased and/or outdated.

Technical Assumption #2 – Data suppied by the US Census Bureau will be used to develop marketing profiles.

## User Stories

**User Story #1**: The UVW marketing team would like general demographic information on the

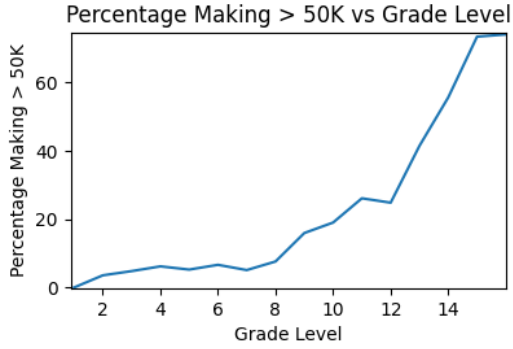
market as a whole. They would like to know the age distribution of people in the census.




The first business objective is to understand the market demographics. Age is a big factor in the enrollment of UVW College and an important demographic to consider. For example, if being 80+ years old is a good indicator for having a high salary, the marketing team may prioritize marketing more expensive programs to this age range. However, based on this histogram, it is clearly shown that a very small percentage of the population is above 80 years old.

I used a histogram (with 20 bins) to depict the age range due to its continuous nature. The majority of people are between 20 to 45. The proportion quickly tapers off after this point. A marketing strategy would reach the most number of people if it targets people under the age of 45.

**User Story #2**: The UVW marketing team would like to know if there is a correlation between a person's level of education and his/her salary.

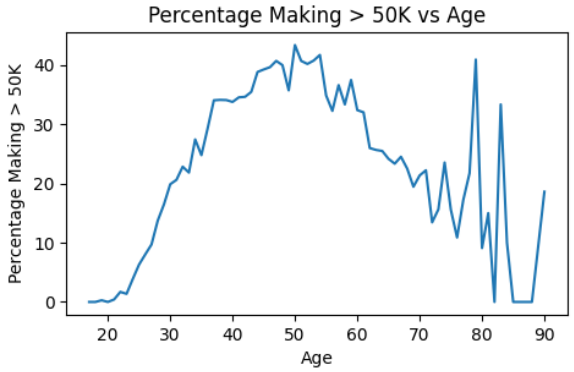



This visualization helps with business objective #2 by created a correlation between grade (or education) level with salary. Grade level is assigned a numeric score, with grades 1 through corresponding to their respective grades in elementary through high school. Grade 13 represents a bachelor's degree, 14 represents a master's degree, 15 presents professional school, and 16 represents a doctorate degree.

To create this visualization, I had to count the number of people at each grade level who made over $50,000 and divide it by the total number of people in that grade level. I then turned this ratio into a percentage and plotted it on a line graph.

This graph shows that the higher the level of education, the more likely that person will make > $50,000.

**User Story #3**: The UVW marketing team would like to know if there is a correlation between a person's age and his/her salary.



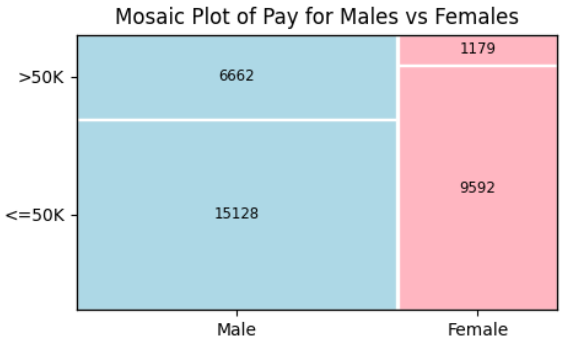

This is similar to user story #2, and helps to show the relationship between age and salary.

To create this visualization, I found the percentage of people making over $50,000 for each age and divided it by the total number of people who are that age. I turned the ratio into a percentage and plotted it vs. age.

This graph shoulds that there is a steady increase of people who make over $50,000 as age

increases until around age 50. Then that ratio steadily decreases. There is variance above the 75 year old mark due to small sample size. This shows that marketing campaigns targeting the middle-age range (around 40-60) should expect about 35-40% of the audience to make over $50,000/year. Those targeting a younger audience (and those that are most likely to attend college) will have much lower salary levels.

**User Story #4**: The UVW marketing team would like to know if there is a correlation between gender and salary.

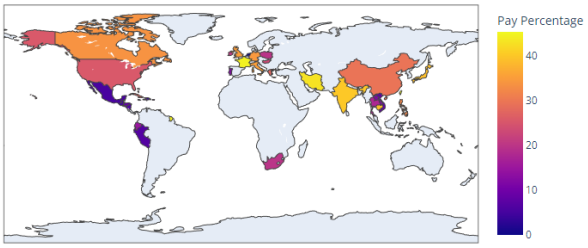



This visualization helps the marketing team predict the effect of gender on pay. It also shows the proportion of genders and pay for each section.

This mosaic plot was created by creating a crosstable to determine the number of males and females who made more or less than $50,000. Then the mosaic graph from statsmodels was used, with colors to easily separate male vs female.

This plot shows that approximately 60% of all respondants were male, and that males were more likely to make at least $50,000. A marketing strategy that targets the female population may want to advertise more economical approaches to attending UVW College.

**User Story #5**: The UVW marketing team would like to know if there is a correlation between native country and salary.

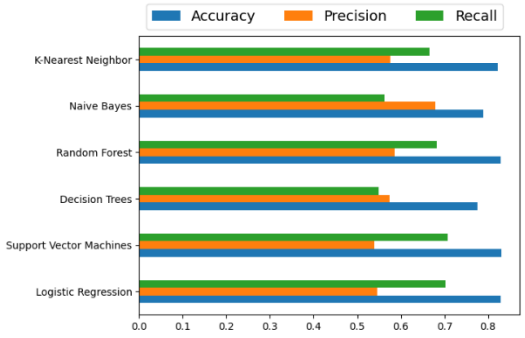



This visualization helps the marketing team easily view the correlation between native country that someone is from and their salary.

This choropleth map was created by first finding the % of people that have a salary of more than $50,000, separated by country. The country names are substituted with ISO 3166 country codes. Then, this data was put into a plotly choropleth map, using a sequential color scheme to represent the percentage of people from that country that make over $50,000 anually.

This plot makes viewing salary from each country easy (a full size image can be found in the appendix). When targeting countries like France, Taiwan, and Iran, it can be expected that the salary of those countries is high. On the other hand, countries like Nicaragua, Peru, or Mexico may have a lower ability to pay for expensive programs.

**User Story #6**: The UVW marketing team would like an algorithm that, based on all the features in the dataset, can predict whether someone has a salary of over or under $50,000.




A variety of binary classifiers were fed training demographic data and the accuracy of the models were calculated using testing data.

I first had to convert categorical data into indicator variables using pandas.get_dummies. For example, an array that looks like [White, Black, Asian-Pac-Islander, White] would be converted into a dataframe that looks like:

| | White | Black | Asian-Pac-Islander |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 |

I used principal component analysis to reduce the dimensionality of the data to 10 components and then fed them into various binary classifiers. The most important statistic was the accuracy of the models, with logistic regression leading the pack at 82% accuracy.

This model can be used to predict, to a reasonable extent, whether someone makes more or less than $50,000. The full python code can be found in the appendix.

## Questions

During the pursuit of this project, I encountered a few challenges/questions:

1. How should I address data with "?" as a value?

Due to the infrequency of these data points, I decided it would be best to simply delete the entire data point. I did so by creating a dataframe of those that had "?" in any of the columns, and then using df.drop() to delete those specific indices.

2. What visualizations would be best to appropriately show the relationship between demographics and pay?

I wanted to show a variety of types of demographics, using continuous variables like age or education level in the form of line plots, binary data like gender in the form of a mosaic plot, or categorical data like native country in the form of a choropleth map. This showed a variety of visualization types and each had clear correlations between the data and pay.

3. How do I put categorical data through a binary classifier?

I found the best solution was to turn categorical data into indicator variables, as explained in user story #6. Although this greatly increased the number of columns in my dataframe, this was mitigated by using principal component analysis to reduce the dimensionality of the data.

4. Which binary classifier was best?

I answered this question by simply using six of the most popular: K-Nearest Neighbor, Naïve Bayes, Random Forest, Decision Trees, Support Vector Machines, and Logistic Regression and then calculating the accuracy of each model. In the end, they all boasted similar accuracies, approximately 80%, with logistic region slightly edging out the competition.

## Not Doing

1. Improve classification model. More time could be spent increasing the accuracy of a model to 90%+
2. Weigh the data based on the student population of UVW College. Finding the correlation between age and pay isn't super useful when the majority of census

participants are outside the normal age of college attendees. In addition, what are the demographics like for UVW student body? What kind of people are most likely to attend?

3. Find correlations between job type, race, marital status, or other demographics and pay.

## Appendix

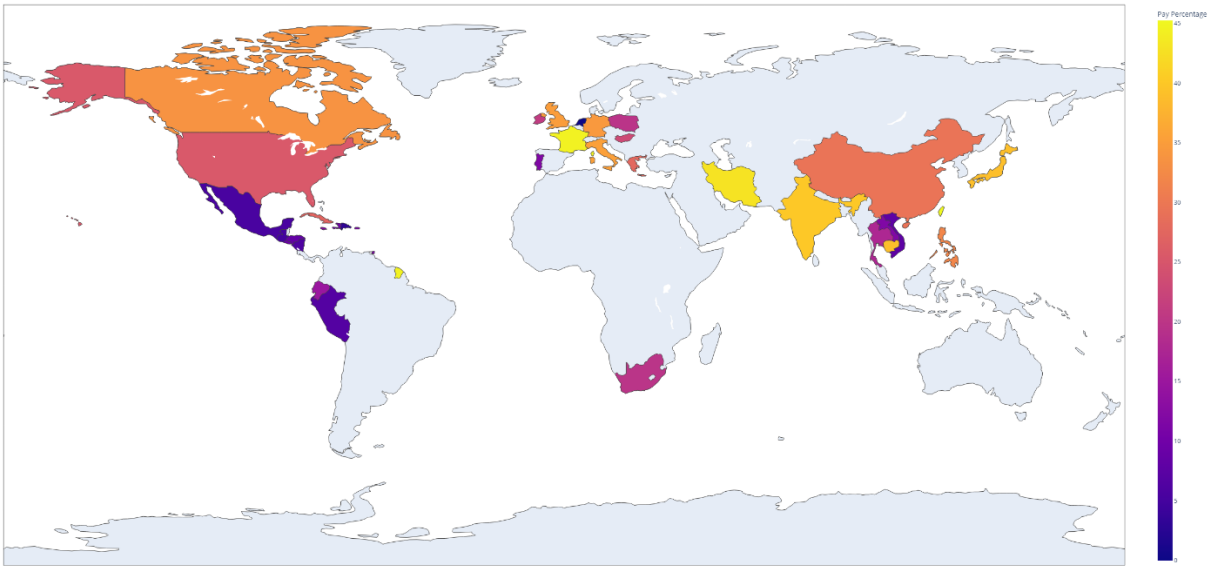



Figure 1. Full-Size Choropleth Map

### Full Python Code – project.py

(run with python3 project.py, with adult.data in the same folder)

```
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn.preprocessing import LabelEncoder
from statsmodels.graphics.mosaicplot import mosaic
from itertools import product
import plotly
import plotly.express as px


###################################
# Importing data into a DataFrame #
#         and cleaning it         #
###################################
```

```
dataList = []

# with open("adult.data", newline="") as csvfile:
#    csvreader = csv.reader(csvfile, delimiter=",", quotechar="|")
#    for row in csvreader:
#        dataList.append(row)


df = pd.read_csv("adult.data", sep=",", header=None, skipinitialspace=True)
df.columns = [
    "Age",
    "Workclass",
    "Fnlwgt",
    "Education",
    "Education Num",
    "Marital Status",
    "Occupation",
    "Relationship",
    "Race",
    "Sex",
    "Capital Gain",
    "Capital Loss",
    "Hours Per Week",
    "Native Country",
    "Pay",
]
# result = df.head(5)
# print(result)

indicesToDelete = df[
    (df["Age"] == "?")
    | (df["Workclass"] == "?")
    | (df["Fnlwgt"] == "?")
    | (df["Education Num"] == "?")
    | (df["Marital Status"] == "?")
    | (df["Occupation"] == "?")
    | (df["Relationship"] == "?")
    | (df["Race"] == "?")
    | (df["Sex"] == "?")
    | (df["Capital Gain"] == "?")
    | (df["Capital Loss"] == "?")
    | (df["Hours Per Week"] == "?")
    | (df["Native Country"] == "?")
    | (df["Pay"] == "?")
].index


df.drop(indicesToDelete, inplace=True)

####################
#   User story 1   #
```

```
# Histogram of Age #
####################
# print("min age: ", min(df["Age"]))
# print("max age: ", max(df["Age"]))
fig, axs = plt.subplots(1, 1, figsize=(5, 4))
axs.hist(df["Age"], bins=20, rwidth=0.9, color="#607c8e")
plt.xlabel("Age")
plt.ylabel("Counts")
plt.title("Histogram of Ages for Census Data")
plt.show()

########################
#     User Story 2     #
# Education vs. Salary #
########################
educationPayDict = {}
educationCountDict = {}
for idx, row in df.iterrows():
    # print(row)
    key = row["Education Num"]
    if row["Education Num"] not in educationCountDict:
        educationCountDict[row["Education Num"]] = 0
        educationPayDict[row["Education Num"]] = 0
    if row["Pay"] == ">50K":
        educationPayDict[row["Education Num"]] += 1

    educationCountDict[row["Education Num"]] += 1

sortedKeys = list(educationPayDict.keys())
sortedKeys.sort()
# print(sortedKeys)
educationPayRatio = {
    i: educationPayDict[i] * 100 / educationCountDict[i] for i in sortedKeys
}
# print(educationPayRatio)
plt.plot(sortedKeys, educationPayRatio.values())
plt.xlabel("Grade Level")
plt.ylabel("Percentage Making > 50K")
plt.title("Percentage Making > 50K vs Grade Level")
plt.show()

##################
#  User Story 3  #
# Age vs. Salary #
##################
agePayDict = {}
ageCountDict = {}
for idx, row in df.iterrows():
    # print(row)
    key = row["Age"]
    if key not in ageCountDict:
```

```
        ageCountDict[key] = 0
        agePayDict[key] = 0
    if row["Pay"] == ">50K":
        agePayDict[key] += 1

    ageCountDict[key] += 1

# print(agePayDict)
# print(ageCountDict)

sortedKeys = list(agePayDict.keys())
sortedKeys.sort()
# print(sortedKeys)
agePayRatio = {i: agePayDict[i] * 100 / ageCountDict[i] for i in sortedKeys}
# print(agePayRatio)
plt.plot(sortedKeys, agePayRatio.values())
plt.xlabel("Age")
plt.ylabel("Percentage Making > 50K")
plt.title("Percentage Making > 50K vs Age")
plt.show()

####################################
#         User Story #4          #
# Mosaic plot of gender vs salary #
####################################

crosstable = pd.crosstab(df["Sex"], df["Pay"])
# print(crosstable)
props = {}
props[("Male", ">50K")] = {"facecolor": "lightblue", "edgecolor": "white"}
props[("Male", "<=50K")] = {"facecolor": "lightblue", "edgecolor": "white"}
props[("Female", ">50K")] = {"facecolor": "lightpink", "edgecolor": "white"}
props[("Female", "<=50K")] = {"facecolor": "lightpink", "edgecolor": "white"}
labelizer = lambda k: {
    ("Male", ">50K"): 6662,
    ("Female", ">50K"): 1179,
    ("Male", "<=50K"): 15128,
    ("Female", "<=50K"): 9592,
}[k]
mosaic(
    df,
    ["Sex", "Pay"],
    labelizer=labelizer,
    properties=props,
    title="Mosaic Plot of Pay for Males vs Females",
)
plt.show()

####################################
#         User Story #5          #
# Choropleth Map of Pay by Country #
```

```
####################################

countryPayDict = {}
countryCountDict = {}
for idx, row in df.iterrows():
    # print(row)
    key = row["Native Country"]
    if key not in countryCountDict:
        countryCountDict[key] = 0
        countryPayDict[key] = 0
    if row["Pay"] == ">50K":
        countryPayDict[key] += 1

    countryCountDict[key] += 1

# print(countryPayDict)
# print(countryCountDict)

sortedKeys = list(countryPayDict.keys())
sortedKeys.sort()
# print(sortedKeys)
countryPayRatio = {i: countryPayDict[i] * 100 / countryCountDict[i] for i in sortedKeys}
del countryPayRatio["Columbia"]
del countryPayRatio["Scotland"]

# print(countryPayRatio)

choroplethDf = pd.DataFrame(
    {
        "Country": [
            "KHM",
            "CAN",
            "CHN",
            "CUB",
            "DOM",
            "ECU",
            "SLV",
            "GBR",
            "FRA",
            "DEU",
            "GRC",
            "GTM",
            "HTI",
            "NLD",
            "HND",
            "HKG",
            "HUN",
            "IND",
            "IRN",
            "IRL",
            "ITA",
```

```
            "JAM",
            "JPN",
            "LAO",
            "MEX",
            "NIC",
            "GUM",
            "PER",
            "PHL",
            "POL",
            "PRT",
            "PRI",
            "ZAF",
            "TWN",
            "THA",
            "TTO",
            "USA",
            "VNM",
            "YUG",
        ],
        "Pay Percentage": list(countryPayRatio.values()),
    }
)
# print(choroplethDf)

fig = px.choropleth(
    choroplethDf,
    locations="Country",
    color="Pay Percentage",
    color_continuous_scale=px.colors.sequential.Plasma,
)
fig.show()

##########################
# Binary Classification #
##########################


df.columns = [c.replace(" ", "_") for c in df.columns]
df["Pay"].replace(["<=50K", ">50K"], [0, 1], inplace=True)
df["Sex"].replace(["Male", "Female"], [0, 1], inplace=True)
# print(df.head(10))

workclassDummies = pd.get_dummies(df.Workclass)
maritalStatusDummies = pd.get_dummies(df.Marital_Status)
occupationDummies = pd.get_dummies(df.Occupation)
raceDummies = pd.get_dummies(df.Race)
nativeCountryDummies = pd.get_dummies(df.Native_Country)
relationshipDummies = pd.get_dummies(df.Relationship)

df = pd.concat(
    [
```

```
        workclassDummies,
        maritalStatusDummies,
        occupationDummies,
        raceDummies,
        nativeCountryDummies,
        relationshipDummies,
        df,
    ],
    axis="columns",
)
df.drop(
    [
        "Workclass",
        "Marital_Status",
        "Occupation",
        "Race",
        "Sex",
        "Native_Country",
        "Education",
        "Relationship",
    ],
    axis=1,
    inplace=True,
)
# print(df.head(10))
# print(df.columns)


X = df.iloc[:, 0:86].values
y = df.iloc[:, 86].values

# print(X[:10])
# print(y[:10])

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Preprocess data to fit standard scale
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Apply PCA function
pca = PCA(n_components=10)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fit Logic Regression to the training set
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

```
# Predicting test set result
y_pred = classifier.predict(X_test)

# Make confusion matrix
TN, FP, FN, TP = confusion_matrix(y_test, y_pred).ravel()
print("True Positive:", TP)
print("False Positive:", FP)
print("True Negative:", TN)
print("False Negative:", FN)

accuracy = (TP + TN) / (TP + FP + TN + FN)
print("Accuracy of the binary classifier = {:0.3f}".format(accuracy))

# Finding performance of various binary classifiers
models = {}

# Logistic Regression
from sklearn.linear_model import LogisticRegression

models["Logistic Regression"] = LogisticRegression()

# Support Vector Machines
from sklearn.svm import LinearSVC

models["Support Vector Machines"] = LinearSVC()

# Decision Trees
from sklearn.tree import DecisionTreeClassifier

models["Decision Trees"] = DecisionTreeClassifier()

# Random Forest
from sklearn.ensemble import RandomForestClassifier

models["Random Forest"] = RandomForestClassifier()

# Naive Bayes
from sklearn.naive_bayes import GaussianNB

models["Naive Bayes"] = GaussianNB()

# K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier

models["K-Nearest Neighbor"] = KNeighborsClassifier()

from sklearn.metrics import accuracy_score, precision_score, recall_score

accuracy, precision, recall = {}, {}, {}
```

```
for key in models.keys():

    # Fit the classifier
    models[key].fit(X_train, y_train)

    # Make predictions
    predictions = models[key].predict(X_test)

    # Calculate metrics
    accuracy[key] = accuracy_score(predictions, y_test)
    precision[key] = precision_score(predictions, y_test)
    recall[key] = recall_score(predictions, y_test)

df_model = pd.DataFrame(
    index=models.keys(), columns=["Accuracy", "Precision", "Recall"]
)
df_model["Accuracy"] = accuracy.values()
df_model["Precision"] = precision.values()
df_model["Recall"] = recall.values()
print(df_model)

ax = df_model.plot.barh()
ax.legend(
    ncol=len(models.keys()), bbox_to_anchor=(0, 1), loc="lower left", prop={"size": 14}
)
plt.tight_layout()
plt.show()
```