Examining COVID-19 Vaccination Rates in California

How are race/ethnicity and income related to vaccination rates?

Manas Khatore, Sahana Noru, Nikki Trueblood, and Sean Yang

```
from datascience import *
import pandas as pd
import seaborn as sns
sns.set_theme(style="ticks", color_codes=True)
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import array
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error
```

Association between race/ethnicity and COVID-19 vaccination rates

```
raw_table = Table().read_table('covid19vaccinesbycountybydemographic.csv')
raw_table
```

county	county_type	demographic_category	demographic
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17
Alameda	MIXED	Age Group	12-17

... (501476 rows omitted)

```
race_table = raw_table.where('demographic_category', 'Race/Ethnicity').drop('county_type', 'demographic_category')
race_table
```

county	demographic_value	est_population
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920
Alameda	American Indian or Alaska Native	4920

... (250638 rows omitted)

The different race/ethnicity categories in "race_table" are American Indian or Alaska Native, Asian, Black or African American, Latino, Multiracial, Native Hawaiian or Other Pacific Islander, Other Race, Unknown and White. Since the table is sorted according to the administration date of the vaccine, we decided to clean the table by only keeping the rows of the most recent administration date (11/12/2021).

```
race_and_county_vax = race_table.where("administered_date", "2021-11-12")
race_and_county_vax
```

county	demographic_value	est_p
Alameda	American Indian or Alaska Native	4920
Alameda	Asian	44218
Alameda	Black or African American	19234
Alameda	Latino	39284
Alameda	Multiracial	71378
Alameda	Native Hawaiian or Other Pacific Islander	13474
Alameda	Other Race	nan
Alameda	Unknown	nan
Alameda	White	56119
Alpine	American Indian or Alaska Native	213

... (521 rows omitted)

```
# adding the percent of population fully vaccinated and with one dose of the vaccine to the table

race_and_county_vax = race_and_county_vax.relabeled('est_age_5plus_pop', 'eligible population')
fully_vax_percent = 100*race_and_county_vax.column('cumulative_fully_vaccinated')/race_and_county_vax.column('eligible population')
one_dose_percent = 100*race_and_county_vax.column('cumulative_unvax_total_pop')/race_and_county_vax.column('eligible population')
race_and_county_vax = race_and_county_vax.with_columns('percent_fully_vaccinated', fully_vax_percent, 'percent_one_dose', one_dose_percent)
race_and_county_vax
```

county	demographic_value	est_p
Alameda	American Indian or Alaska Native	4920
Alameda	Asian	44218
Alameda	Black or African American	19234
Alameda	Latino	39284
Alameda	Multiracial	71378
Alameda	Native Hawaiian or Other Pacific Islander	13474
Alameda	Other Race	nan
Alameda	Unknown	nan
Alameda	White	56119
Alpine	American Indian or Alaska Native	213

... (521 rows omitted)

Looking at the table, we noticed that many values in the "Other Race" and "Unknown" categories are nan. Checking the percentages in the table that are over 100:

```
above_100 = race_and_county_vax.where('percent_fully_vaccinated', are.above(100))
above_100.show()
```

county	demographic_value
Amador	Native Hawaiian or Other Pacific Islander
Contra Costa	Native Hawaiian or Other Pacific Islander
El Dorado	Native Hawaiian or Other Pacific Islander
Kern	Native Hawaiian or Other Pacific Islander
Los Angeles	Native Hawaiian or Other Pacific Islander
Marin	Native Hawaiian or Other Pacific Islander
Mendocino	Native Hawaiian or Other Pacific Islander
Napa	Native Hawaiian or Other Pacific Islander
Placer	Native Hawaiian or Other Pacific Islander
San Benito	Native Hawaiian or Other Pacific Islander
San Diego	Native Hawaiian or Other Pacific Islander
San Joaquin	Native Hawaiian or Other Pacific Islander
San Luis Obispo	Native Hawaiian or Other Pacific Islander
Santa Barbara	Native Hawaiian or Other Pacific Islander
Santa Clara	Native Hawaiian or Other Pacific Islander
Santa Cruz	Native Hawaiian or Other Pacific Islander
Solano	Native Hawaiian or Other Pacific Islander
Trinity	Asian
Tulare	Native Hawaiian or Other Pacific Islander
Tuolumne	Native Hawaiian or Other Pacific Islander

Since the "Native Hawaiian or Other Pacific Islander" demographic largely contains percentages that are over 100, we decided to fully remove it from the table, as well as removing "Other Race" and "Unknown."

```
# removing demographics and percentages higher than 100
# converting to a DataFrame for visualization purposes

final_race_and_county_vax = race_and_county_vax.where('demographic_value', are.not_equal_to('Native Hawaiian or Other Pacific Islander')).wh
final_race_and_county_vax = final_race_and_county_vax.select("county", "demographic_value", "est_population", "eligible population", "cumula
final_race_and_county_vax_df = final_race_and_county_vax.to_df()
final_race_and_county_vax_df
```

	county object Alameda 1.8% Butte 1.8% 57 others 96.5%	demographic_value object Latino 17.4% White 17.4% 4 others 65.3%	est_population float64 39.0 – 15796762.0	eligible populati… float… 37.0 – 14779309.0	cumulative_fully_va… floa… 8.0 – 8740833.0	cumulative_at_least… floa… 11.0 – 9534525.0
0	Alameda	American Indian or Alaska Native	4920	4607	3464	3883
1	Alameda	Asian	442180	420851	393032	418970
2	Alameda	Black or African American	192346	183187	95089	103167
3	Alameda	Latino	392842	369205	213925	232837
4	Alameda	Multiracial	71378	64602	27683	29464
Expand rows 5 - 334						
335	Yuba	Asian	5549	5158	2758	2987
336	Yuba	Black or African American	2359	2149	1084	1220

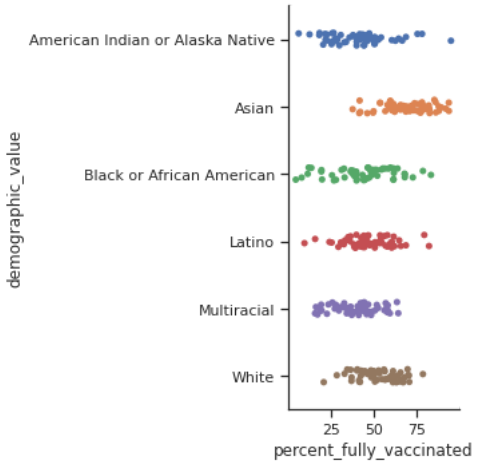
	county object	demographic_value object	est_population float64	eligible populati… float…	cumulative_fully_va… floa…	cumulative_at_least… floa…
	Alameda 1.8%	Latino 17.4%	39.0 – 15796762.0	37.0 – 14779309.0	8.0 – 8740833.0	11.0 – 9534525.0
	Butte 1.8%	White 17.4%				
	57 others 96.5%	4 others 65.3%				
337	Yuba	Latino	20874	19088	8411	9433
338	Yuba	Multiracial	3682	3321	781	841
339	Yuba	White	44110	41222	17527	19179

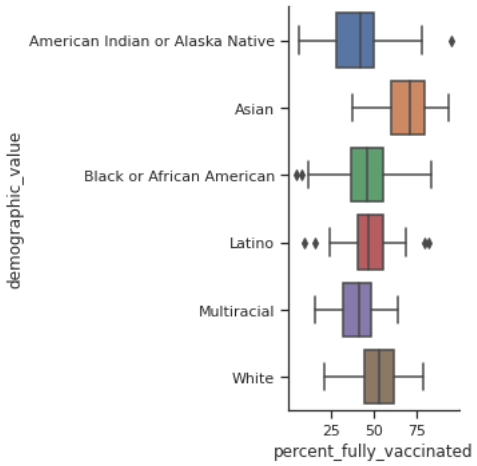
340 rows × 8 columns

```
%matplotlib inline
```

```
sns.catplot(x="percent_fully_vaccinated", y="demographic_value", data=final_race_and_county_vax_df)
# black dots/diamonds in the boxplot represent outliers
sns.catplot(x='percent_fully_vaccinated', y='demographic_value', kind='box', data=final_race_and_county_vax_df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdf32bd9710>
```





Seeing the differences in fully vaccinated rates along the different demographic groups, we decided to perform an A/B test to determine whether certain groups were getting vaccinated at statistically significantly higher rates than others. Instead of analyzing the differences in each combination of groups, we decided to split the demographics into "traditionally represented" and "traditionally underrepresented" racial groups, using sources about presence in higher education.

```
represented = final_race_and_county_vax.where('demographic_value', are.not_equal_to("American Indian or Alaska Native")).where('demographic_
represented = represented.group('county', sum)
new_percent_full = represented.column('cumulative_fully_vaccinated sum') / represented.column('eligible population sum') * 100
new_percent_one_dose = represented.column('cumulative_at_least_one_dose sum') / represented.column('eligible population sum') * 100
represented = represented.drop('percent_fully_vaccinated sum', 'percent_one_dose sum').with_columns('percent_fully_vaccinated', new_percent_
labels = []
for i in range(represented.num_rows):
    labels = np.append(labels, 'represented')
represented = represented.with_column('label', labels)
represented
```

county	est_population sum	eligible population sum
Alameda	1.07475e+06	1.01928e+06
Alpine	791	774
Amador	29917	28739
Butte	183347	173980
Calaveras	36694	35199
Colusa	9051	8569
Contra Costa	750054	715066
Del Norte	19067	18070
El Dorado	159481	152923
Fresno	423830	396120

... (48 rows omitted)

```
underrepresented = final_race_and_county_vax.where('demographic_value', are.not_equal_to("Asian")).where('demographic_value', are.not_equal_
underrepresented = underrepresented.group('county', np.sum)
new_percent_full = underrepresented.column('cumulative_fully_vaccinated sum') / underrepresented.column('eligible population sum') * 100
new_percent_one_dose = underrepresented.column('cumulative_at_least_one_dose sum') / underrepresented.column('eligible population sum') * 10
underrepresented = underrepresented.drop('percent_fully_vaccinated sum', 'percent_one_dose sum').with_columns('percent_fully_vaccinated', ne
labels = make_array()
for i in np.arange(underrepresented.num_rows):
    labels = np.append(labels, "underrepresented")
underrepresented = underrepresented.with_column('label', labels)
underrepresented
```

county	est_population sum	eligible population sum
Alameda	590108	556999
Alpine	303	291
Amador	6989	6675
Butte	43101	40307
Calaveras	6053	5686
Colusa	13673	12721
Contra Costa	404179	379414
Del Norte	7744	7376
El Dorado	29349	27500
Fresno	596578	551054

... (48 rows omitted)

A/B Testing

Null: In the total population, the distribution of percent vaccinated is the same for underrepresented racial groups as it is for other racial groups.

Alternative: In the total population, the distribution of percent vaccinated is lower for underrepresented racial groups than for other racial groups.

Test Statistic: (average percent of represented people who are fully vaccinated) - (average percent of underrepresented people who are vaccinated)

```
# we shuffle the labels and define the function test_stat() that will calculate test statistics
# we calculate the observed test statistic from the data

represented_df = represented.to_df()
underrepresented_df = underrepresented.to_df()
frames = [represented_df, underrepresented_df]
rep_vs_underrep = pd.concat(frames)
shuffled_labels = rep_vs_underrep.sample(n = 58)['label']

def test_stat(tbl):
    represented_values = tbl.loc[(tbl['label'] == 'represented')]
    underrepresented_values = tbl.loc[(tbl['label'] == 'underrepresented')]
    return represented_values['percent_fully_vaccinated'].mean() - underrepresented_values['percent_fully_vaccinated'].mean()

observed_difference = test_stat(rep_vs_underrep)
observed_difference
```

```
8.012822377438376
```

```
# one simulation of the A/B test

def one_simulated_difference():
    shuffled_labels = rep_vs_underrep.sample(n = 116)['label']
    shuffled_table = rep_vs_underrep.drop(columns = ['label'])
    shuffled_table['label'] = shuffled_labels.tolist()
    return test_stat(shuffled_table)

one_simulated_difference()
```

```
-2.68538865736231
```

```
# completing 10000 simulations and storing the results in an array

differences = []
for i in range(10000):
    new_difference = one_simulated_difference()
    differences = np.append(differences, new_difference)
differences_df = pd.DataFrame(differences, columns = ['Differences'])
differences_df
```

	Differences float64 -9.263564033901375 – 9.279…
0	-4.485241518612092
1	1.1607280745292528
2	-1.4365790349003973
3	-0.40294433575559196
4	1.1027912596484555
Expand rows 5 - 9994	
9995	0.5221857374315064
9996	-1.1362501912725307
9997	0.670555790938316
9998	-3.119982803264996

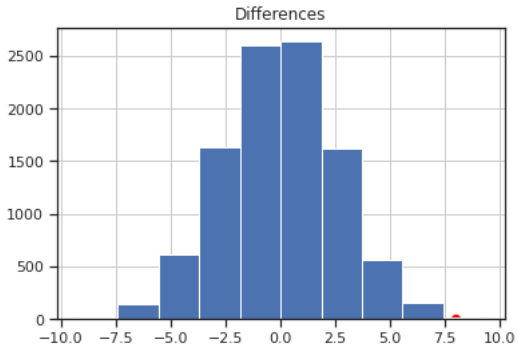
Differences float64
-9.263564033901375 – 9.279…

9999	0.7482272806328538

10000 rows × 1 columns

```
# plotting the results of the simulations
# the red dot represents the observed test statistic

differences_df.hist()
plt.scatter(observed_difference, -0.002, color='red', s=40);
```



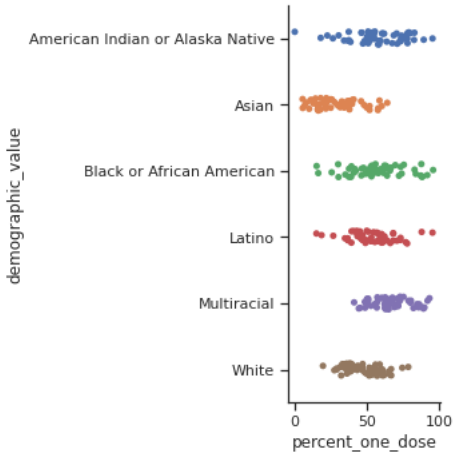

```
above_observed = differences_df.loc[differences_df['Differences'] >= observed_difference]
above_observed.astype(bool).sum(axis=0)/10000
```

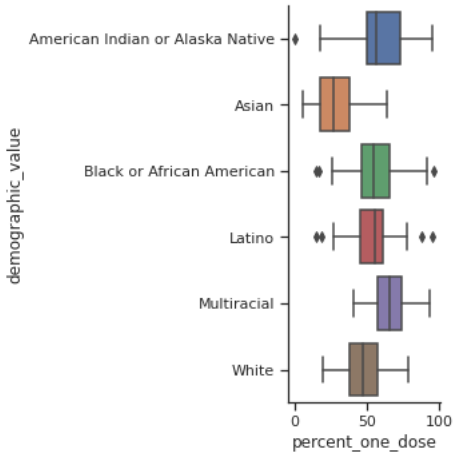
```
Differences    0.0011
dtype: float64
```

Our calculated p-value is 0.0011. Using a standard p-value cutoff of 0.05, we would reject the null hypothesis. From our analysis, we would conclude that traditionally underrepresented groups are being vaccinated a lower rates than other groups in California.

```
sns.catplot(x="percent_one_dose", y="demographic_value", data=final_race_and_county_vax_df)
# black dots/diamonts represent outliers
sns.catplot(x='percent_one_dose', y='demographic_value', kind='box', data=final_race_and_county_vax_df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdf31a70b50>
```





The above visualizations show that some groups that had low rates of full vaccination have high rates of one dose vaccination, such as the American Indian/Alaska Native and Multiracial demographics. This can potentially be explained by these groups getting vaccinated in larger numbers more recently, or that members in these groups only received one does and did not return for the second.

Linear Regression between COVID-19 vaccination rates and median income per county

```
# importing a table that contains the number of people who are fully vaccinated or received at least one dose per county

vaccines = Table().read_table('covid19vaccinesbycounty.csv')
new_vaccines = vaccines.where('county', are.not_equal_to('All CA Counties')).where('county', are.not_equal_to('All CA and Non-CA Counties'))
new_vaccines = new_vaccines.group('county', sum).select('county', 'fully_vaccinated sum', 'at_least_one_dose sum')
new_vaccines
```

county	fully_vaccinated sum	at_least_one_dose sum
Alameda	1209773	1304079
Alpine	704	810
Amador	19220	22518
Butte	105405	116000
Calaveras	22572	25735
Colusa	11561	12845
Contra Costa	852735	910645
Del Norte	11711	13299
El Dorado	107366	117966
Fresno	529697	596413

... (48 rows omitted)

```
# importing a table that contains the median income per county in 2018
# adding a column with the total population in each county
# using the total population and the new_vaccines table to calculate the percent of fully vaccinated and at least one dose per county

income_table = Table().read_table('incomebycounty.csv')
int_values = income_table.apply(lambda x: int((x[0:x.index(',')] + x[x.index(',') + 1:])), 'Value')
income_table = income_table.drop('Value').with_column("Median income", int_values)
population_array = make_array(1682253, 1204, 40474, 211632,45292,21839,1165927, 27743,191185,1008654,28917,136463, 179702, 19016,909235,1524
income_table = income_table.with_column('Total population', population_array)
full_vax = 100*new_vaccines.column('fully_vaccinated sum')/income_table.column('Total population')
one_dose = 100*new_vaccines.column('at_least_one_dose sum')/income_table.column('Total population')
income_table = income_table.with_columns('Fully vaccinated percentage', full_vax, 'At least one dose percentage', one_dose)
income_table_df = income_table.to_df()
income_table_df
```

	County object Alameda ..... 1.7% Alpine ..... 1.7% 56 others. 96.6%	Median income int64 38497 – 116178	Total population int64 1204 – 10014009	Fully vaccinated pe… floa… 24.225481209899176 – 77.95…	At least one dose p… floa… 27.115795905896732 – 88.51…
0	Alameda	92574	1682253	71.91385600144568	77.51979042391365
1	Alpine	64688	1204	58.47176079734219	67.27574750830564
2	Amador	61198	40474	47.4872757819835	55.63571675643623
3	Butte	48443	211632	49.80579496484464	54.81212671051637
4	Calaveras	58151	45292	49.83661573787865	56.82018899584916
Expand rows 5 - 52					
53	Tulare	47518	473117	45.907883250866064	52.12875462095
54	Tuolumne	56493	55620	47.41999280834232	53.403451995685
55	Ventura	84017	843843	65.16603206994667	70.96391153330656
56	Yolo	65923	216403	63.531466754157755	70.98931160843426
57	Yuba	52624	81575	41.21605884155685	45.78240882623353

58 rows × 5 columns

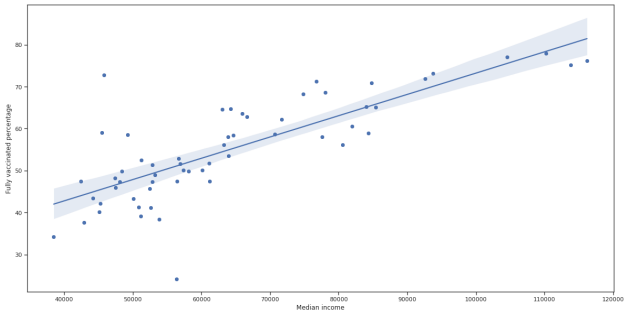
```
# plotting fully vaccinated percentage vs. median income

fig, ax = plt.subplots(figsize = (20,10))
plt.scatter(data = income_table_df, x = "Median income", y = "Fully vaccinated percentage")

plt.xlabel("Median income")
plt.ylabel("Fully vaccinated percentage")
income_and_fully_vax = income_table_df[["Median income", "Fully vaccinated percentage"]]
sns.regplot(income_and_fully_vax['Median income'], income_and_fully_vax['Fully vaccinated percentage'])

print("The correlation coefficient is: " + str(income_and_fully_vax.corr(method ='pearson').iloc[0, 1]))
```

```
/shared-libs/python3.7/py/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. F
  FutureWarning
The correlation coefficient is: 0.7889963375011697
```



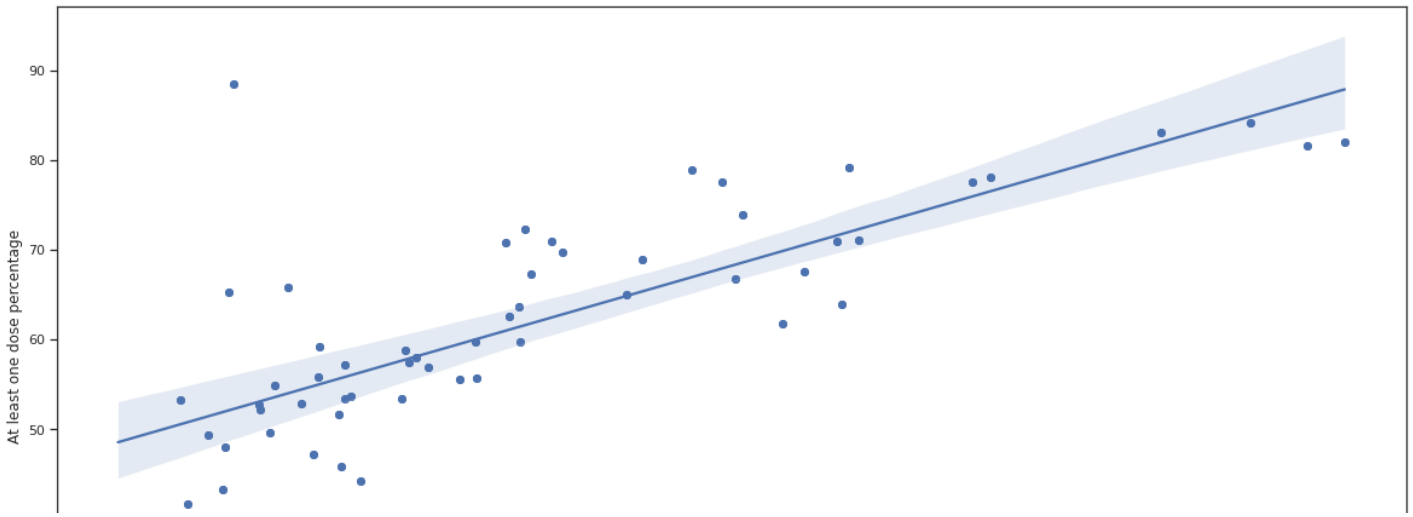
```
# plotting at least one dose percentage vs. median income

fig, ax = plt.subplots(figsize = (20,10))
plt.scatter(data = income_table_df, x = "Median income", y = "At least one dose percentage")

plt.xlabel("Median income") # You need to manually add these in!
plt.ylabel("At least one dose percentage")
income_and_one_dose = income_table_df[["Median income", "At least one dose percentage"]]
sns.regplot(income_and_one_dose['Median income'], income_and_one_dose['At least one dose percentage'])

print("The correlation coefficient is: " + str(income_and_one_dose.corr(method ='pearson').iloc[0, 1]))
```

```
/shared-libs/python3.7/py/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. Fr
  FutureWarning
The correlation coefficient is: 0.7431907440294258
```



The graphs above show that there is a strong, positive correlation between median income and the percentage of people who are fully vaccinated or have at least one dose in each county. Seeing that there is a correlation, we decided to create linear regression models that would be able to predict the percentage of vaccinated people given the median income of an area.

```
# Linear model predicting fully vaccinated percentage from median income

train_data, test_data = train_test_split(income_table_df, test_size = 0.20, random_state = 50)

X_train = train_data[['Median income']]
y_train = train_data['Fully vaccinated percentage']

X_test = test_data[['Median income']]
y_test = test_data['Fully vaccinated percentage']

model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("The coefficient of determination is " + str(metrics.r2_score(y_test, y_pred)))
print("The mean squared error is " + str(mean_squared_error(y_test, y_pred)))
```

```
The coefficient of determination is 0.7402790757964923
The mean squared error is 25.511459853252248
```

```
# Linear model predicting one dose percentage from median income

X_train = train_data[['Median income']]
y_train = train_data['At least one dose percentage']

X_test = test_data[['Median income']]
y_test = test_data['At least one dose percentage']
```

```
model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("The coefficient of determination is " + str(metrics.r2_score(y_test, y_pred)))
print("The mean squared error is " + str(mean_squared_error(y_test, y_pred)))
```

```
The coefficient of determination is 0.6936244616227263
The mean squared error is 28.336745087480566
```

The coefficient of determination shows the proportion of outcomes that can be explained by the model. In order to reduce the mean squared error for the models, we decided to standardize the units by converting each measure to the number of standard deviations it is away from its mean.

```
standardized_income = (income_table_df["Median income"]-income_table_df["Median income"].mean())/income_table_df["Median income"].std()
standardized_full_vax = (income_table_df["Fully vaccinated percentage"]-income_table_df["Fully vaccinated percentage"].mean())/income_table_
standardized_one_dose = (income_table_df["At least one dose percentage"]-income_table_df["At least one dose percentage"].mean())/income_tabl
standards = {"Median income": standardized_income, "Fully vaccinated percentage": standardized_full_vax, "At least one dose percentage": sta
standardized_income_table_df = pd.DataFrame(data=standards)
```

```
train_data, test_data = train_test_split(standardized_income_table_df, test_size = 0.20, random_state = 50)

X_train = train_data[['Median income']]
y_train = train_data['Fully vaccinated percentage']

X_test = test_data[['Median income']]
y_test = test_data['Fully vaccinated percentage']

model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("The coefficient of determination is " + str(metrics.r2_score(y_test, y_pred)))
print("The mean squared error is " + str(mean_squared_error(y_test, y_pred)))
```

```
The coefficient of determination is 0.7402790757964932
The mean squared error is 0.17516638882496108
```

```
X_train = train_data[['Median income']]
y_train = train_data['At least one dose percentage']

X_test = test_data[['Median income']]
y_test = test_data['At least one dose percentage']

model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("The coefficient of determination is " + str(metrics.r2_score(y_test, y_pred)))
print("The mean squared error is " + str(mean_squared_error(y_test, y_pred)))
```

```
The coefficient of determination is 0.6936244616227262
The mean squared error is 0.17316482475617587
```