```
from sklearn.model_selection import train_test_split

X = data.drop(['median_house_value'], axis =1)
y = data['median_house_value']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

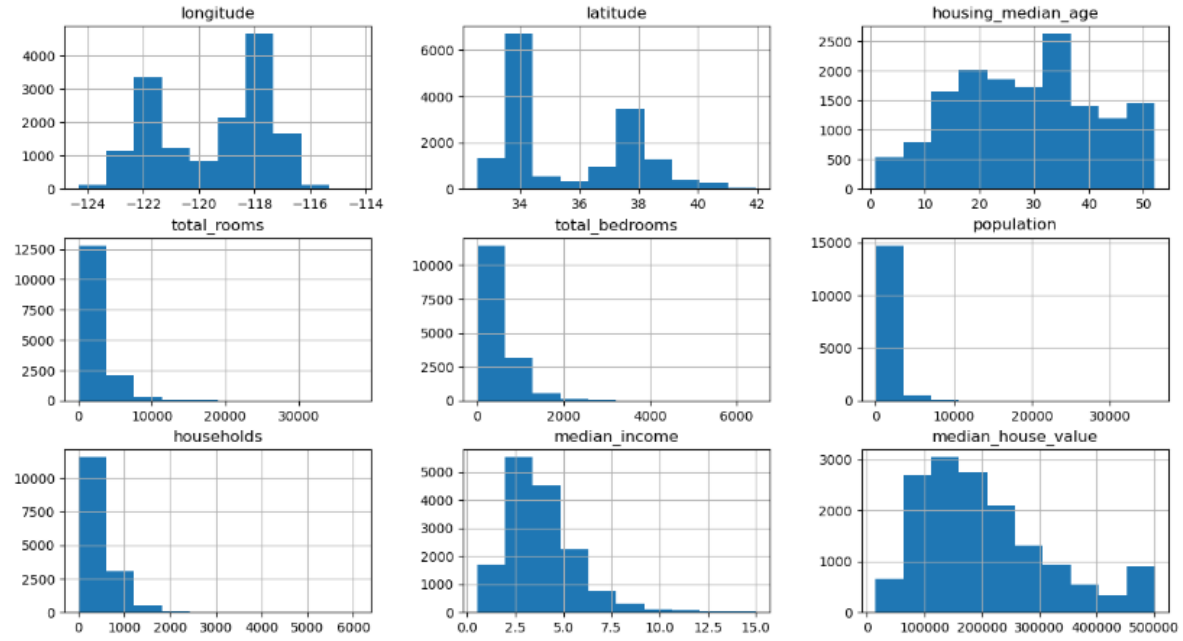
```
train_data = X_train.join(y_train)
```

```
train_data
```

...

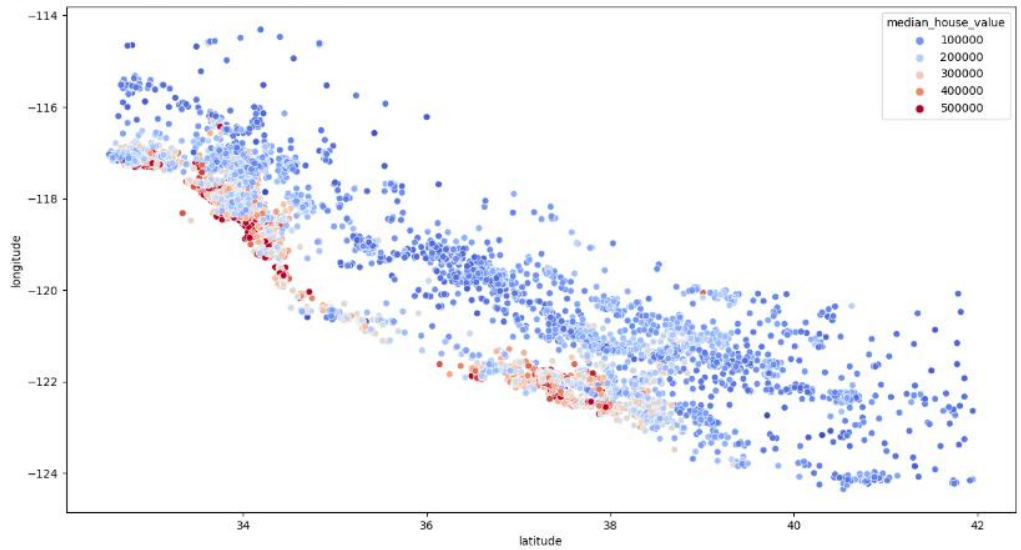
```
train_data.hist(figsize=(15,8))
```

```
array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>],
       [<Axes: title={'center': 'total_rooms'}>,
        <Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>],
       [<Axes: title={'center': 'households'}>,
        <Axes: title={'center': 'median_income'}>,
        <Axes: title={'center': 'median_house_value'}>]], dtype=object)
```




```
plt.figure(figsize=(15,8))
sns.scatterplot(x="latitude", y="longitude", data = train_data, hue="median_house_value", palette="coolwarm")
```

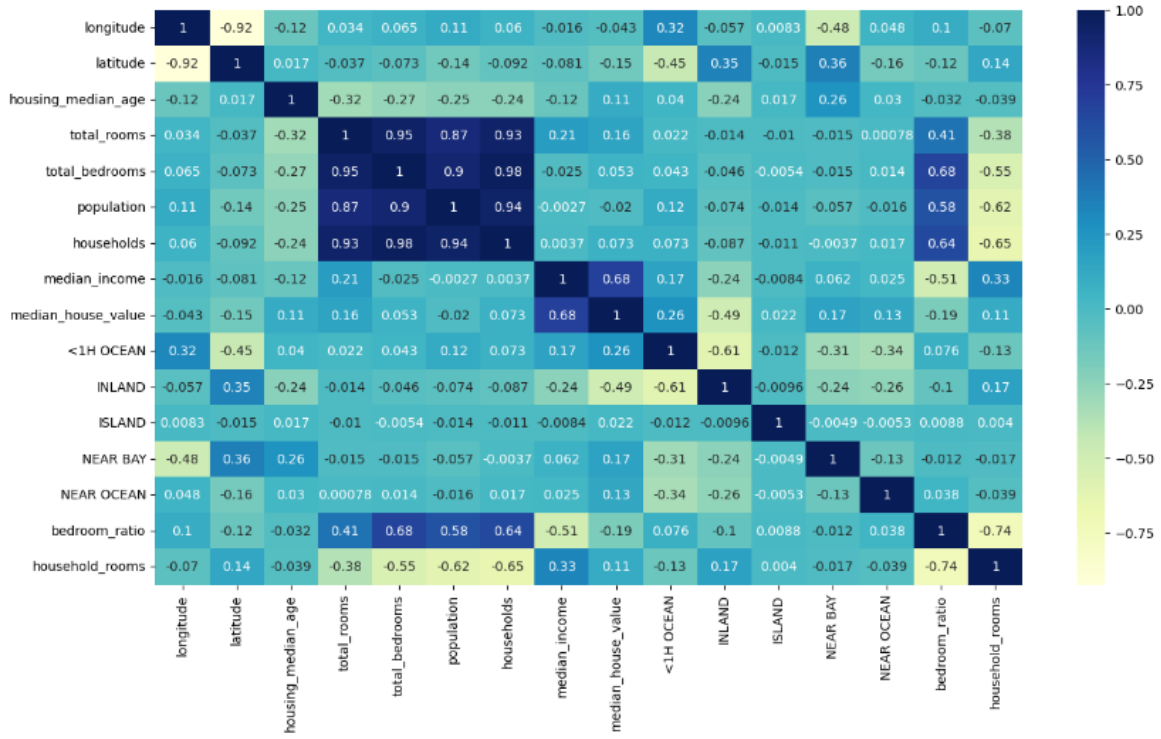
```
<Axes: xlabel='latitude', ylabel='longitude'>
```



```
#Feauture Engineering
train_data['bedroom_ratio'] = train_data['total_bedrooms']/ train_data['total_rooms']
train_data['household_rooms'] = train_data['total_rooms']/train_data['households']
```

```
plt.figure(figsize=(15,8))
sns.heatmap(train_data.corr(), annot=True, cmap='YlGnBu')
```

```
<Axes: >
```




```
from sklearn.linear_model import LinearRegression

X_train, y_train = train_data.drop(['median_house_value'], axis =1), train_data['median_house_value']

reg = LinearRegression()
reg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
test_data = X_test.join(y_test)

#Data Preprocessing
test_data['total_rooms'] = np.log(test_data['total_rooms']+1)
test_data['total_bedrooms'] = np.log(test_data['total_bedrooms']+1)
test_data['population'] = np.log(test_data['population']+1)
test_data['households'] = np.log(test_data['households']+1)

test_data = test_data.join(pd.get_dummies(test_data.ocean_proximity)).drop(['ocean_proximity'], axis =1) #Made the different str

#Feauture Engineering
test_data['bedroom_ratio'] = test_data['total_bedrooms']/ test_data['total_rooms']
test_data['household_rooms'] = test_data['total_rooms']/test_data['households']
```

```
X_test, y_test = test_data.drop(['median_house_value'], axis =1), test_data['median_house_value']
```

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

```
0.673389329293721
```

```
#Random Forest & Hyper Parameter Tuning
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor()
forest.fit(X_train, y_train)
```

RandomForestRegressor

RandomForestRegressor()

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

0.8222769738969176

```
from sklearn.model_selection import GridSearchCV

forest = RandomForestRegressor()

param_grid = {
    "n_estimators": [3,10,30],
    "max_features": [2,4,6,8]
}

grid_search = GridSearchCV(forest, param_grid, cv=5,
                           scoring="neg_mean_squared_error",
                           return_train_score=True)

grid_search.fit(X_train, y_train)
```

GridSearchCV

estimator: RandomForestRegressor

RandomForestRegressor

```
best_forest = grid_search.best_estimator_
```

```
best_forest
```

RandomForestRegressor

RandomForestRegressor(max_features=8, n_estimators=30)

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

0.8183072898131833