Client Report - Can you predict that?

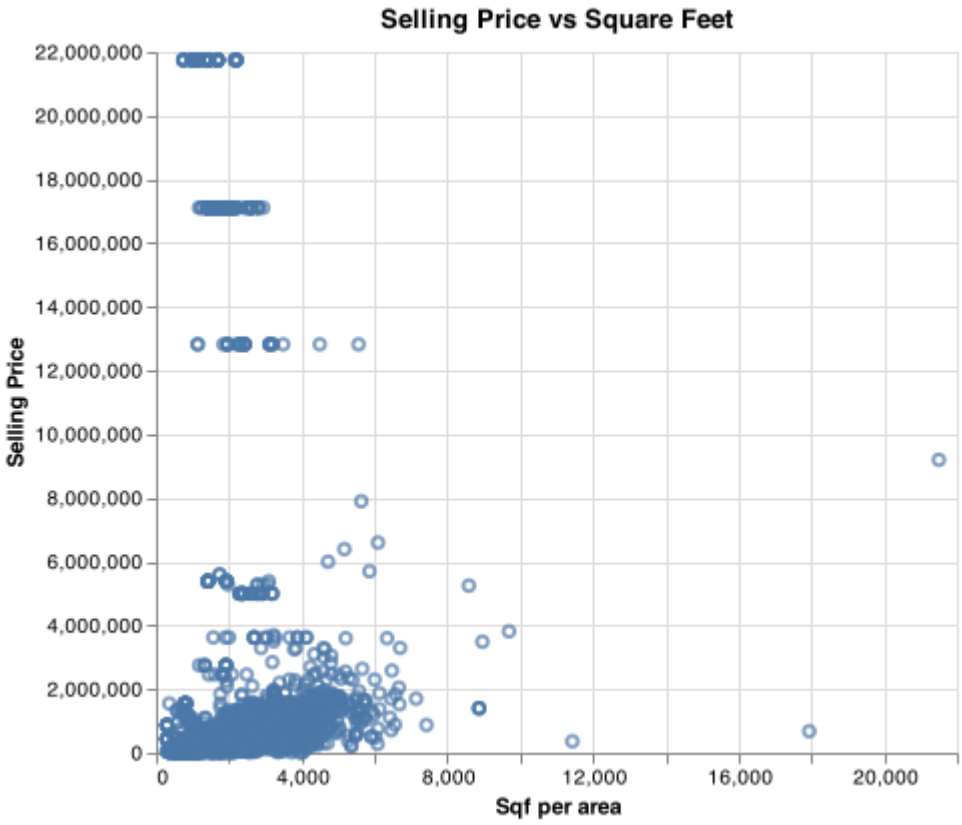
Course CSE 250 Isabel Aranguren

Elevator pitch

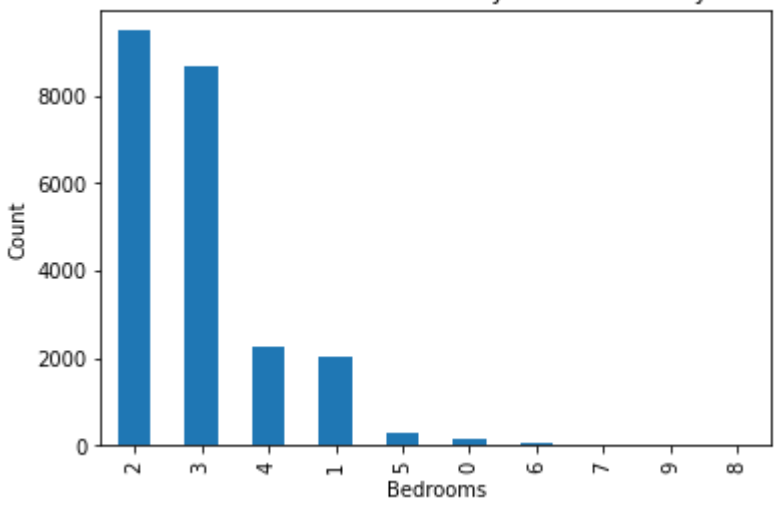
The state of Colorado has a big chunk of its residential housing data missing the year of construction, and they'd like to design a prediction model that can classify whether a house was built before 1980. They'd also like to create a model that predicts (regression) each home's actual age.

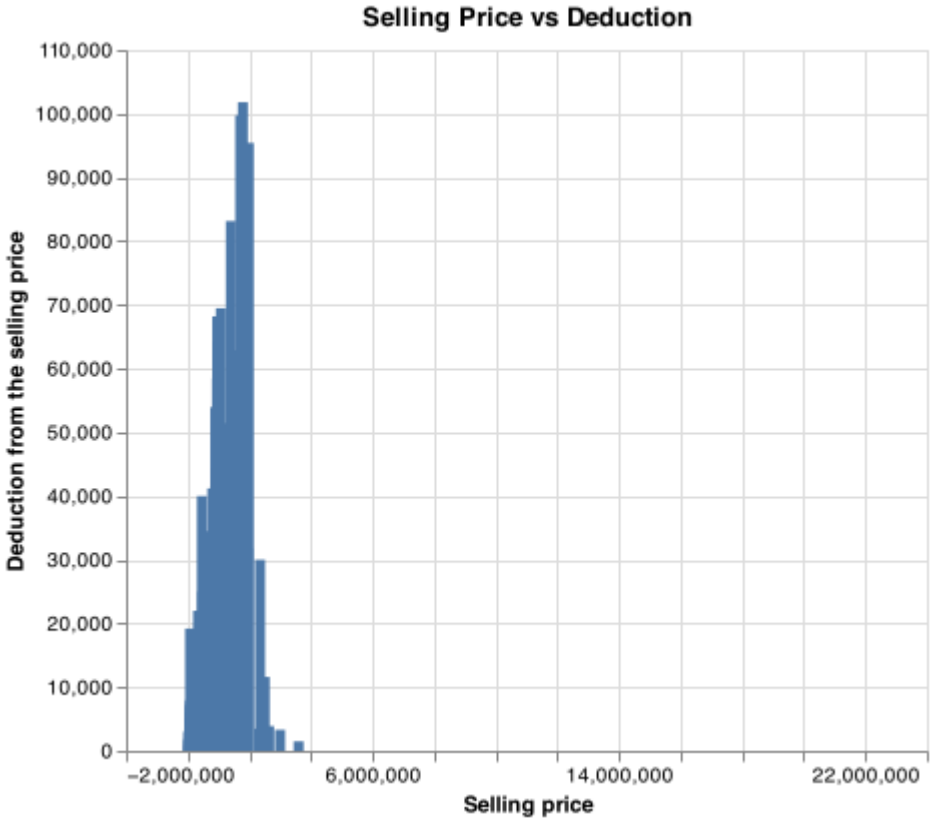
GRAND QUESTION 1

Create 2-3 charts that evaluate potential relationships between the home variables and before1980



2 bedroom houses are most commonly sold followed by 3 bedroom





TECHNICAL DETAILS

```
# %%
chart = (alt.Chart(dwellings_denver, title="Selling Price vs Square Feet")
    .encode(
        alt.X("livearea:Q", title = "Sqf per area"),
        alt.Y('sprice:Q',title="Selling Price")
    )
    .mark_point().properties(width=400,height=350)
)
# %% [markdown]
# Second Chart: Selling Price vs Deduction
# %%
chart = (alt.Chart(dwellings_denver, title="Selling Price vs Deduction")
    .encode(
        alt.X("sprice:Q", title = "Selling price"),
        alt.Y('deduct:Q',title="Deduction from the selling price")
    )
    .mark_bar().properties(width=400,height=350)
)
chart
# chart.save("price.png")
# %% [markdown]
# 2 bedroom houses are most commonly sold followed by 3 bedroom
# %%
dwellings_denver['numbdrm'].value_counts().plot(kind='bar')
plt.title(' 2 bedroom houses are most commonly sold followed by 3
bedroom')
```

```
plt.xlabel('Bedrooms')
plt.ylabel('Count')
sns.despine
```

GRAND QUESTION 2

Can you build a classification model (before or after 1980) that has at least 90% accuracy for the state of Colorado to use (explain your model choice and which models you tried)?

Yes, I can build a clasiffication model that has 90% accuracy. This is the only model that I got working.

TECHNICAL DETAILS

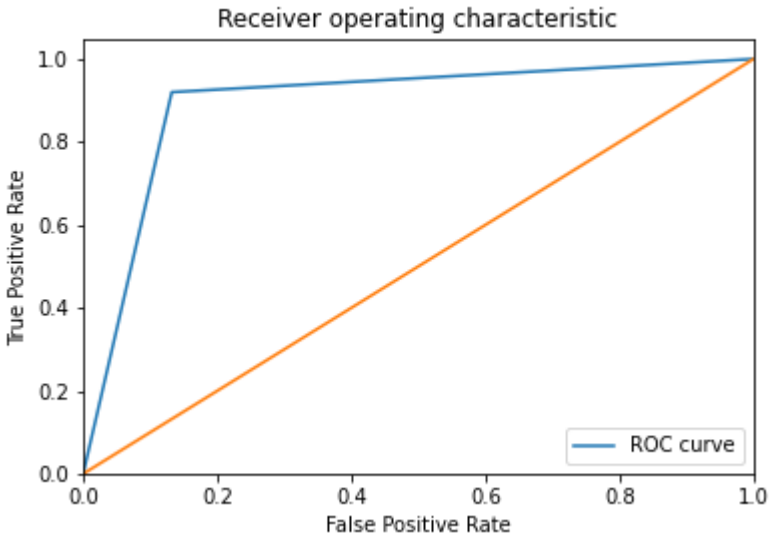
The quality of data that goes towards training the model determines the quality of the output model. The data4dwelings data contains a lot of redundant features, that need to be removed, in order to train a robust model.

```
# Removes the target and keeps all features
#split dataset in features and target variable

X_pred = dwellings_ml.drop(dwellings_ml.filter(regex =
'before1980|yrbuilt|parcel').columns, axis = 1)
# Selects the target column
y_pred = dwellings_ml.filter(regex = "before1980")
# Splitting X and y variables into train and test sets using stratified
sampling
X_train, X_test, y_train, y_test = train_test_split(
    X_pred, y_pred, test_size = .34, random_state = 76)

# clf stands for classifier model.
clf = tree.DecisionTreeClassifier()
# Train Decision Tree Classifer
clf = clf.fit(X_train, y_train)
#  Using the features in the test set to make predictions
y_pred = clf.predict(X_test)
y_probs = clf.predict_proba(X_test)
```

Below is the output of the functions. To read the ROC curve, we are looking to have the bend of the curve in the upper left corner. The closer the curve is to the 45-degree line, the worse our model performs.



GRAND QUESTION 3

Will you justify your classification model by detailing the most important features in your model (a chart and a description are a must?

The most important feature is "arcstyle" also known as the type of home

TECHNICAL DETAILS

```
df_features = pd.DataFrame(
    {'f_names': X_train.columns,
    'f_values': clf.feature_importances_}).sort_values('f_values',
ascending = False)
```

.png)

```
df_features.plot.bar(x='f_names', y='f_values', rot=90, width=.9,figsize=
(20,10), title="Feature Importance Ranking")
```

GRAND QUESTION 4

Can you describe the quality of your classification model using 2-3 evaluation metrics? You need to provide an interpretation of each evaluation metric when you provide the value.

The best value is 1 and the worst value is 0.

```
# %%
print(metrics.classification_report(y_test, y_test_score))
# output Accuracy: 0.9050186112180721
```

Output

precision	recall	f1-score	support	
0	0.89	0.86	0.87	2982
1	0.91	0.93	0.92	4809
accuracy	0.91	7791		
macro avg	0.90	0.90	0.90	7791
weighted avg	0.90	0.91	0.90	7791

APPENDIX A (PYTHON CODE)

```
# %%
import pandas as pd
import altair as alt
import numpy as np
import altair as alt

# %%
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree
Classifier
from sklearn.metrics import (roc_curve, auc, roc_auc_score,
                             confusion_matrix)
import matplotlib.pyplot as plt
import numpy as np
import itertools

# %%
# Load data
dwellings =
pd.read_csv('https://github.com/byuidatascience/data4dwellings/raw/master/
data-raw/dwellings_denver/dwellings_denver.csv')
dwellings_ml =
pd.read_csv('https://github.com/byuidatascience/data4dwellings/raw/master/
data-raw/dwellings_ml/dwellings_ml.csv')
neighborhood =
pd.read_csv('https://github.com/byuidatascience/data4dwellings/raw/master/
data-raw/dwellings_neighborhoods_ml/dwellings_neighborhoods_ml.csv')
alt.data_transformers.enable('data_server')

# %%
# Removes the target and keeps all features
#split dataset in features and target variable
X = dwellings_ml.drop(dwellings_ml.filter(regex =
'before1980|yrbuilt|parcel').columns, axis = 1)
# Selects the target column
```

```
y = dwellings_ml.filter(regex = "before1980")
# Splitting X and y variables into train and test sets using stratified
sampling
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = .33,
random_state = 77)

# %%
# clf stands for classifier model.
clf = DecisionTreeClassifier()
# Train Decision Tree Classifer
clf.fit(X_train, y_train)
#  Using the features in the test set to make predictions
y_train_hat = clf.predict(X_train)
y_test_hat = clf.predict(X_test)

# %%
def get_auc_scores(dec_tree, X_train, X_test, y_train, y_test):
    y_train_score = dec_tree.predict_proba(X_train)[:, 1]
    y_test_score = dec_tree.predict_proba(X_test)[:, 1]
    auc_train = roc_auc_score(y_train, y_train_score)
    auc_test = roc_auc_score(y_test, y_test_score)
    print(f"""
    Training AUC: {auc_train}
    Testing AUC: {auc_test}""")
    return y_test_score
y_test_score = get_auc_scores(clf, X_train, X_test, y_train, y_test)

# %% [markdown]
# Below is the output of this function. To read the ROC curve, we are
looking to have the bend of the curve in the upper left corner. The closer
the curve is to the 45-degree line, the worse our model performs.
#

# %%
def plot_roc_curve(y_test, y_test_score):
    # fpr is the false positive rate, tpr is the true positive rate.
    fpr, tpr, _ = roc_curve(y_test, y_test_score)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    plt.plot(fpr, tpr, label="ROC curve")
    plt.plot([0, 1], [0, 1])
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Receiver operating characteristic")
    plt.legend(loc="lower right")
    plt.savefig('roc_curve.png')
    plt.show()
plot_roc_curve(y_test, y_test_score)

# %%
df_features = pd.DataFrame(
    {'f_names': X_train.columns,
```

```
    'f_values': clf.feature_importances_}).sort_values('f_values',
ascending = False)

# %%
print(metrics.classification_report(y_test, y_test_score))

# %%
print("Accuracy:",metrics.accuracy_score(y_test, y_test_score))
```