Predicting Wine Choices

Project Summary:

This project aims to predict the type of wine a user would like to drink, given the data that is presented to us. Using Machine Learning, we'd classified potential types of wine that a consumer would choose. The dataset was found on Kaggle.

Motivation:

As ML engineers, we believe in automating processes to make lives easier for businesses. One of the factors for which we chose to address this problem was the fact that the wine-making process is tedious. Breweries make lots of wine, but being able to predict which wines are most effective via ML makes their job easier!

1.0: Load Dataset

In [1]:
```
# Imports
from collections import defaultdict
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import plotly.express as px
import seaborn as sns
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import os
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
import warnings
```

In [2]:
```
raw_data = pd.read_csv('winemag-data-130k-v2.csv')
```

In [3]:
```
# Ignore all warnings
warnings.filterwarnings('ignore')
```

1.1: Data Exploration

In [4]:
```
#lines of CSV file because actual file is too large
df = raw_data.head(15000)
```

In [5]:
```
# Get column names
df.columns
```

Out[5]:
```
Index(['id', 'country', 'description', 'designation', 'points', 'price',
       'province', 'region_1', 'region_2', 'taster_name',
       'taster_twitter_handle', 'title', 'variety', 'winery'],
      dtype='object')
```

In [6]:
```
# Check how many NaN values there are
df.isna().sum()
```

Out[6]:
```
id                          0
country                     8
description                 0
designation              4334
points                      0
price                    1055
province                    8
region_1                 2533
region_2                 9170
taster_name              3071
taster_twitter_handle    3615
title                       0
variety                     0
winery                      0
dtype: int64
```

In [7]:
```
df.nunique()
```

Out[7]:
```
id                       15000
country                     37
description              14889
designation               7658
points                      21
price                      192
province                   273
region_1                   822
region_2                    17
taster_name                 18
taster_twitter_handle       14
title                    14865
variety                    390
winery                    6992
dtype: int64
```

1.3: Data Cleaning + Preprocessing

Preprocessing Columns

In [8]:
```
# Drop columns, set index, drop NaNs and duplicates
df.drop(columns=["taster_name", "taster_twitter_handle"], inplace=True)
df = df.set_index('id')
df.dropna(axis=0, inplace=True)
df.drop_duplicates(inplace=True)
df = df.reset_index(drop=True)
```

In [9]:
```
df.head(5)
```

Out[9]:

	country	description	designation	points	price	province	region_1	region_2	title	va
0	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot
1	US	Soft, supple plum envelopes an oaky structure ...	Mountain Cuvée	87	19.0	California	Napa Valley	Napa	Kirkland Signature 2011 Mountain Cuvée Caberne...	Cabe Sauvi
2	US	This wine from the Geneseo district offers aro...	Signature Selection	87	22.0	California	Paso Robles	Central Coast	Bianchi 2011 Signature Selection Merlot (Paso ...	M
3	US	Oak and earth intermingle around robust aromas...	King Ridge Vineyard	87	69.0	California	Sonoma Coast	Sonoma	Castello di Amorosa 2011 King Ridge Vineyard P...	Pinot
4	US	Rustic and dry, this has flavors of berries, c...	Puma Springs Vineyard	86	50.0	California	Dry Creek Valley	Sonoma	Envolve 2010 Puma Springs Vineyard Red (Dry Cr...	Red E

In [10]:
```
# Check shape to check number of entries available
df.shape
```

Out[10]:
```
(3907, 11)
```

Natural Language Processing

In [11]:
```
# Get stop words and add any other common ones from text file into the set
stopWords = set(stopwords.words('english'))

# Remove \n characters
with open('Common English Words.txt') as file:
    cleaned_cwords = [word[:len(word)-1] for word in file.readlines()]

# Add text file common words to stop words set
for c_words in cleaned_cwords:
    stopWords.add(c_words)

# Remove wine variety names in description list by adding variety names to
for wine_variety in df['variety']:
    stopWords.add(wine_variety)
```

In [12]:
```
lemmatizer = WordNetLemmatizer()

# Take the description column and take out stop words
processed_sentences = []
for desc in df['description']:
    preprocessed = [word.lower().strip() for word in desc.split() if (word.

    # Perform regex to remove digits and % signs from list
    for index, word in enumerate(preprocessed):
        match = re.search("[\d%,.]",word)

        # Check if there is a Match object and remove that item from list
        if match is not None:
            preprocessed.pop(index)

    # Lemmatized words in preprocessed list to prevent duplicate word count
    preprocessed = [lemmatizer.lemmatize(word) for word in preprocessed]

    # Combine all cleaned words into string for storage
    processed_sentences.append(" ".join(preprocessed))
```

In [13]:
```
# Store as new column in dataframe
for i in range(df.shape[0]):
    df.loc[i,"processed_description"] = processed_sentences[i]
```

In [14]:
```
# Standardize the price and points columns
scaler = StandardScaler()
standardize_col = ['points', 'price']
for col in standardize_col:
    df[f"{col}_std"] = scaler.fit_transform(df[col].values.reshape(-1,1))
```

In [15]:
```
df.head()
```

Out[15]:

	country	description	designation	points	price	province	region_1	region_2	title	va
0	US	Much like the regular bottling from 2012, this...	Vintner's Reserve Wild Child Block	87	65.0	Oregon	Willamette Valley	Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	Pinot
1	US	Soft, supple plum envelopes an oaky structure ...	Mountain Cuvée	87	19.0	California	Napa Valley	Napa	Kirkland Signature 2011 Mountain Cuvée Caberne...	Cabe Sauvi
2	US	This wine from the Geneseo district offers aro...	Signature Selection	87	22.0	California	Paso Robles	Central Coast	Bianchi 2011 Signature Selection Merlot (Paso ...	M
3	US	Oak and earth intermingle around robust aromas...	King Ridge Vineyard	87	69.0	California	Sonoma Coast	Sonoma	Castello di Amorosa 2011 King Ridge Vineyard P...	Pinot
4	US	Rustic and dry, this has flavors of berries, c...	Puma Springs Vineyard	86	50.0	California	Dry Creek Valley	Sonoma	Envolve 2010 Puma Springs Vineyard Red (Dry Cr...	Red E

1.3.1: Feature Engineering/Encoding

In [16]:
```
# Create new copy of original df to store mapped values
df_mapped = df.copy()
```

In [17]:
```
# Map the columns with numeric variables for PCA
for mapp_kw in ['winery', 'province', 'variety']:
    mapped = {value:index for index,value in enumerate(list(df[mapp_kw].uni
    df_mapped[mapp_kw].replace(list(mapped.keys()), list(mapped.values()),
```

In [18]:
```
# Drop original description column
df = df.drop(['description'],axis=1)
df_mapped = df_mapped.drop(['description'],axis=1)
```

In [19]:
```
# View the last 5 values to check mapping
df_mapped.tail(5)
```

Out[19]:

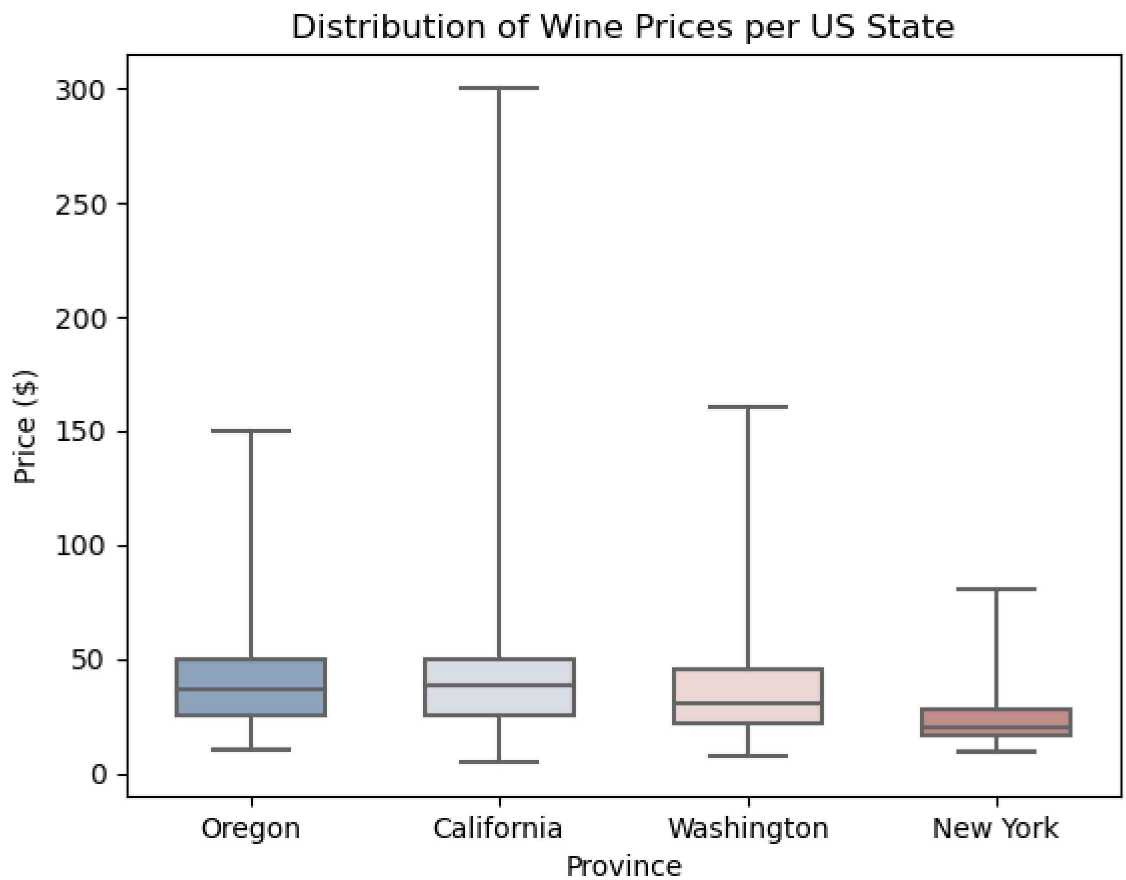
	country	designation	points	price	province	region_1	region_2	title	variety	winery
3902	US	Estate	88	29.0	1	Sierra Foothills	Sierra Foothills	Naggiar 2009 Estate Petite Sirah (Sierra Footh...	8	650
3903	US	Radieux	88	31.0	2	Columbia Valley (WA)	Columbia Valley	Ott & Murphy 2008 Radieux Red (Columbia Valley...	7	1355
3904	US	Dry	88	16.0	3	Finger Lakes	Finger Lakes	Silver Thread 2011 Dry Riesling (Finger Lakes)	12	363
3905	US	The Illusionist	88	45.0	2	Columbia Valley (WA)	Columbia Valley	Sleight of Hand 2009 The Illusionist Red (Colu...	3	1021
3906	US	Anna's Vineyard Version	88	36.0	1	Paso Robles	Central Coast	Adelaida 2009 Anna's Vineyard Version Red (Pas...	14	1161

2.0: Data Visualization

In [20]:
```
# Define absolute path for saving figures
save_path = os.path.abspath('Charts')
```

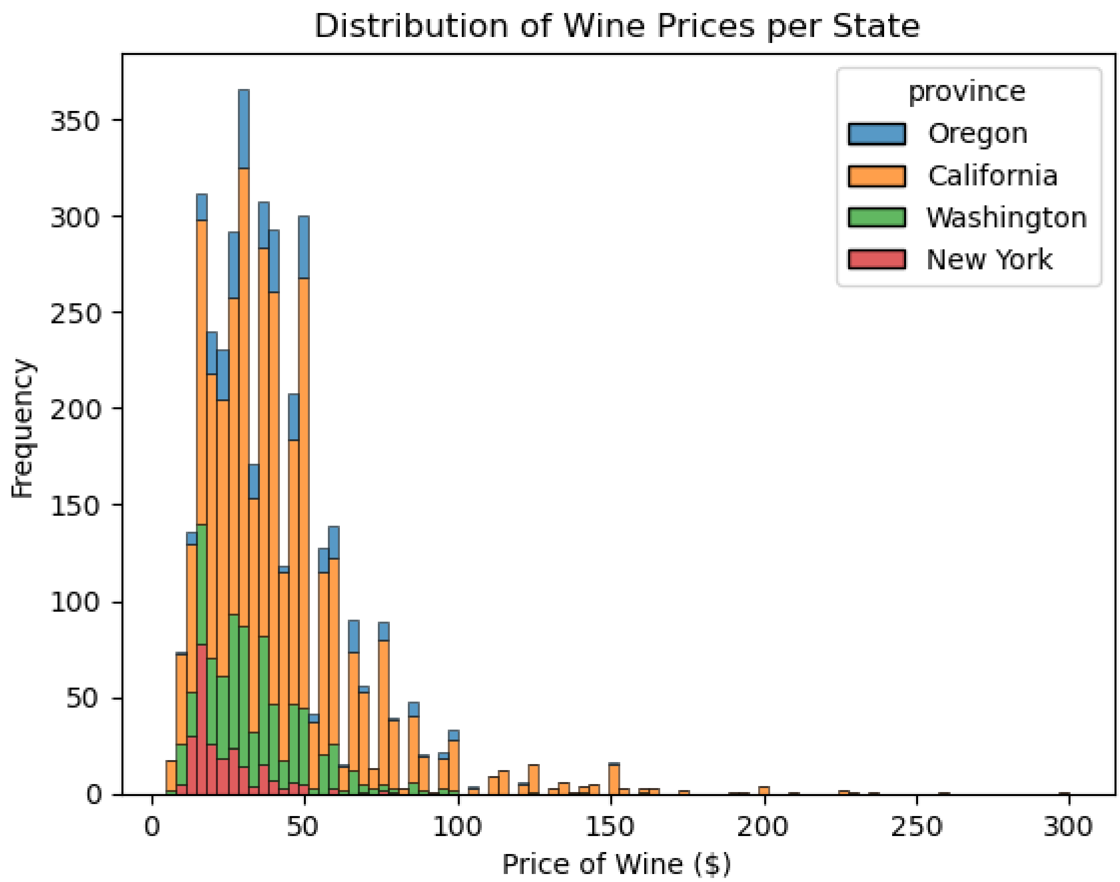
In [21]:
```
# Boxplot relationship between points and province
sns.boxplot(x="province", y="price", data=df, whis=[0, 100],
width=.6, palette="vlag").set(title='Distribution of Wine Prices per US Sta
                                            xlabel='Province',
                                            ylabel='Price ($)');

# Save figure
plt.savefig(save_path + '/WinePrices_perState_BXP.jpg')
```

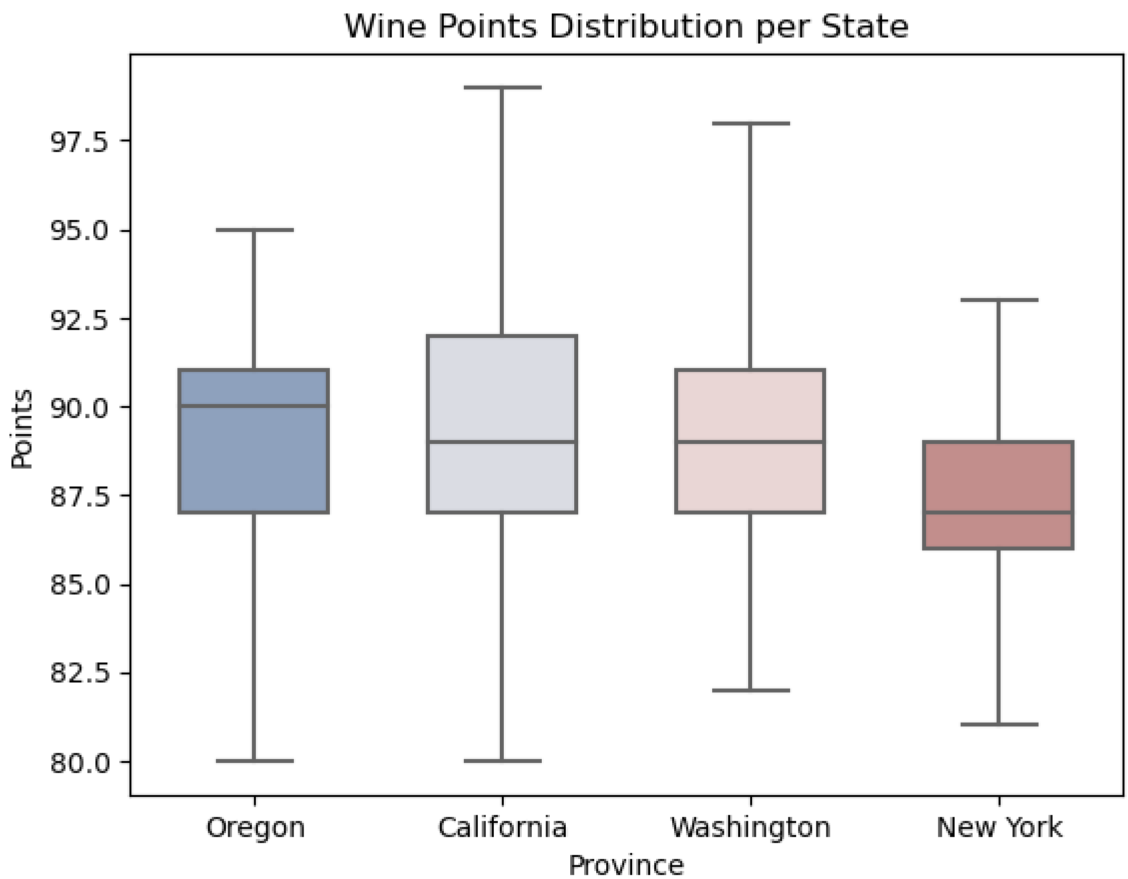


In [22]:
```
sns.histplot(df, x='price',hue='province', multiple='stack').set(title='Dis
                                                                 xlabel='Pr
                                                                 ylabel='Fr

# Save fig
plt.savefig(save_path + '/PriceperProvince_HIST.jpg')
```

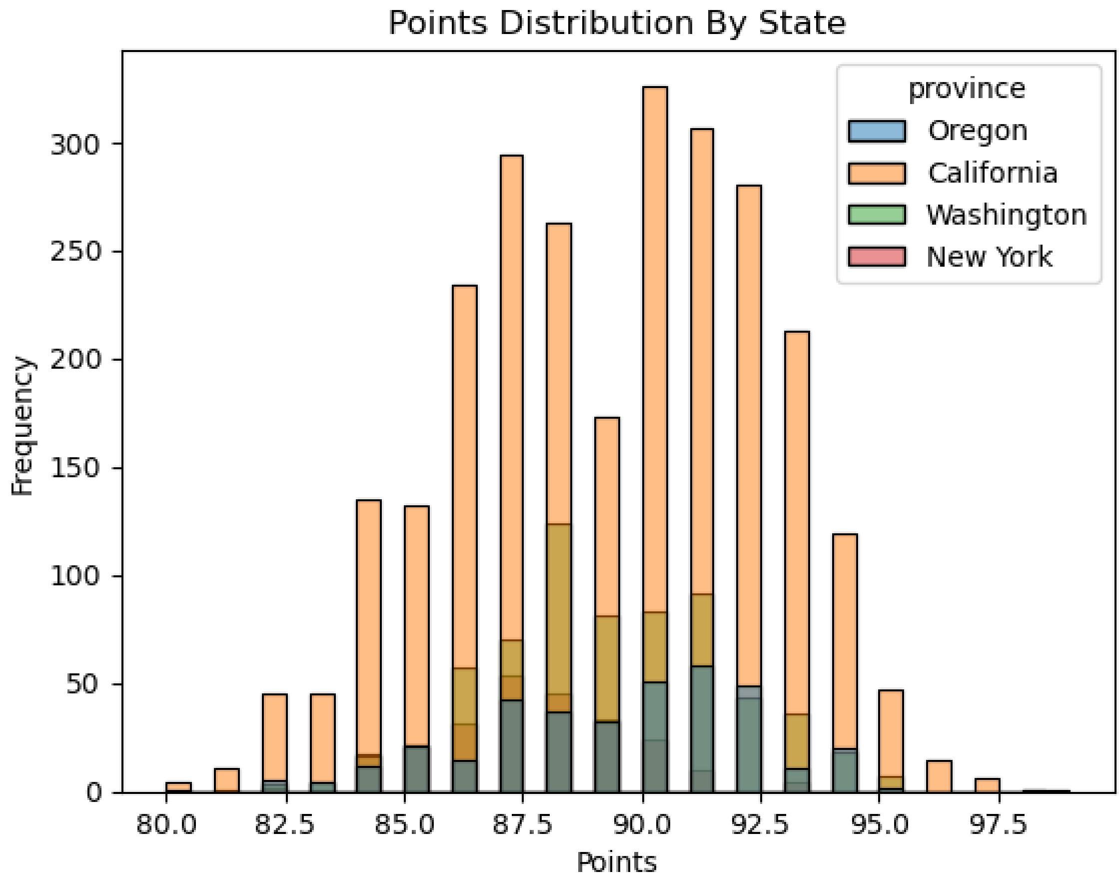


In [23]:
```
# Boxplot relationship between points and province
sns.boxplot(x="province", y="points", data=df, whis=[0, 100],
            width=.6, palette="vlag").set(title='Wine Points Distribution p
                                    xlabel='Province', ylabel='Points
# Save fig
plt.savefig(save_path + '/PointsperProvince_BXP.jpg')
```



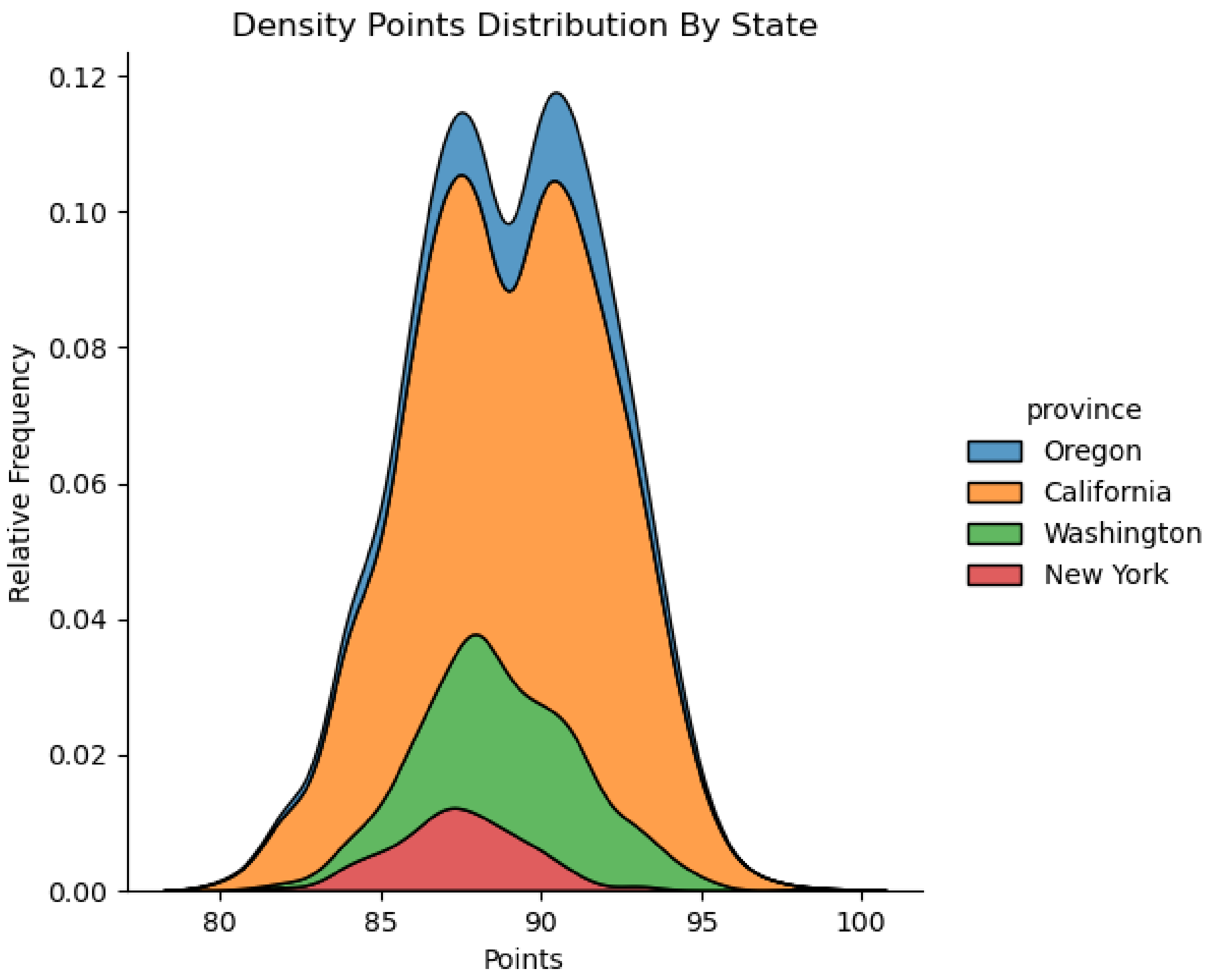
In [24]:
```
sns.histplot(df, x='points',hue='province').set(title='Points Distribution
                                                            xlabel='P

# Save fig
plt.savefig(save_path + '/PointsperProvince_HIST.jpg')
```



In [25]:
```
sns.displot(df, x="points", hue='province', kind="kde", multiple="stack").s
                                                                xlabel='P

# Save fig
plt.savefig(save_path + '/PointDensityperState_DIST.jpg')
```



In [26]:
```
# Count number of times a word occures in all descriptions for all wine bot
def_dict = defaultdict(lambda: 0)
for sentences in df['processed_description']:
    for word in sentences.split():
        def_dict[word] = def_dict[word] + 1
```

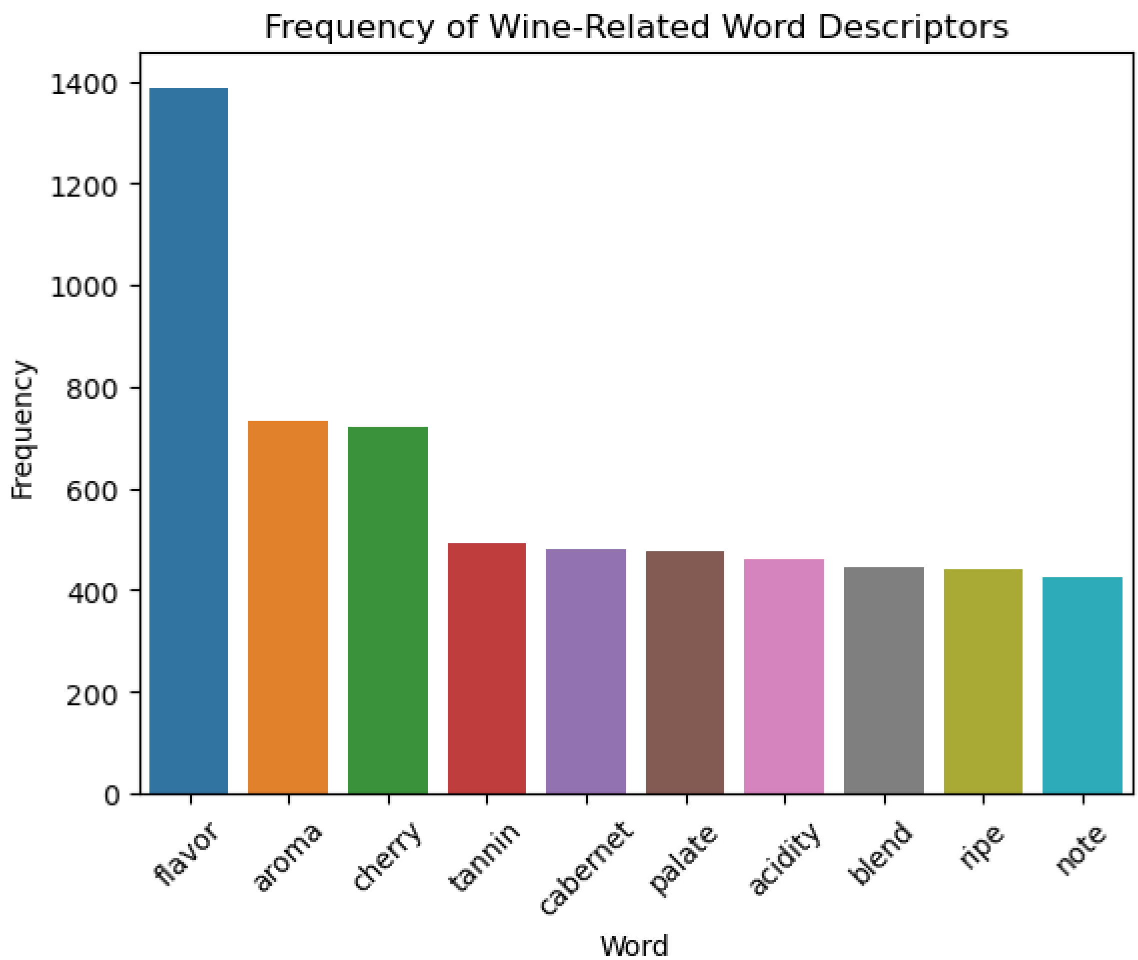
In [27]:
```
# Create df for word counts and sort in descending order by frequency. Stor
word_counts = pd.DataFrame()
word_counts['Words'] = def_dict.keys()
word_counts['Frequency'] = def_dict.values()
word_counts = word_counts.sort_values('Frequency', ascending=False)
word_counts = word_counts[:10]
word_counts
```

Out[27]:

	Words	Frequency
33	flavor	1386
29	aroma	734
36	cherry	722
95	tannin	491
59	cabernet	479
195	palate	475
34	acidity	462
107	blend	444
87	ripe	442
64	note	427

In [28]:
```
# Plot frequency of top 10 word occurences
sns.barplot(data=word_counts, x="Words", y="Frequency")
plt.xticks(rotation=45);
plt.title('Frequency of Wine-Related Word Descriptors')
plt.xlabel('Word');

# Save fig
plt.savefig(save_path + '/DescriptionFrequency_BAR.jpg')
```



In [29]:
```
df_mapped.head(5)
```

Out[29]:

	country	designation	points	price	province	region_1	region_2	title	variety	winery	pı
0	US	Vintner's Reserve Wild Child Block	87	65.0	0	Willamette Valley	Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child...	0	0	
1	US	Mountain Cuvée	87	19.0	1	Napa Valley	Napa	Kirkland Signature 2011 Mountain Cuvée Caberne...	1	1	
2	US	Signature Selection	87	22.0	1	Paso Robles	Central Coast	Bianchi 2011 Signature Selection Merlot (Paso ...	2	2	
3	US	King Ridge Vineyard	87	69.0	1	Sonoma Coast	Sonoma	Castello di Amorosa 2011 King Ridge Vineyard P...	0	3	(
4	US	Puma Springs Vineyard	86	50.0	1	Dry Creek Valley	Sonoma	Envolve 2010 Puma Springs Vineyard Red (Dry Cr...	3	4	I

3.0: Dimensionality Reduction (PCA) for Cluster/Grouping

In [30]:
```
# Perform PCA with quantative features to group
x_feat_list = ['points', 'price', 'winery', 'province', 'variety']
x_vals = df_mapped.loc[:,x_feat_list].values
```

In [31]:
```
# Use PCA to fit and transform the features
pca = PCA()
pca.fit_transform(x_vals)
```

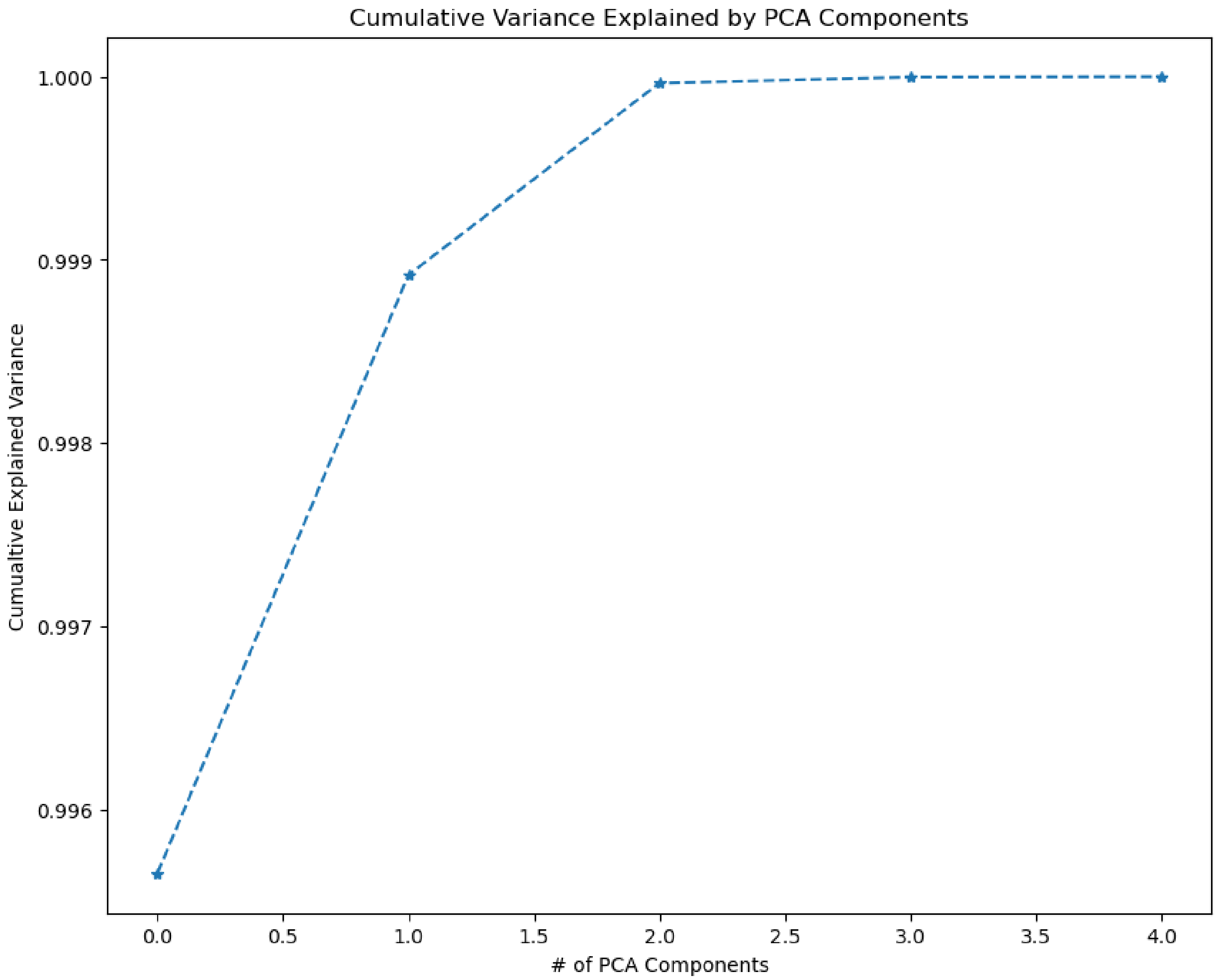
Out[31]:
```
array([[-6.62028840e+02,  2.46855630e+01, -5.18666746e+00,
         3.65708253e+00, -1.10482971e+00],
       [-6.60981314e+02, -2.08125627e+01, -1.16012582e+01,
         1.20548014e+00, -2.88791426e-01],
       [-6.59982757e+02, -1.80155919e+01, -1.01328133e+01,
         1.35863513e+00, -2.81609161e-01],
       ...,
       [-2.98962459e+02, -2.50364649e+01, -1.71573734e+00,
        -1.83804931e-01,  1.67426292e+00],
       [ 3.58993314e+02,  5.82852021e+00, -6.84018295e+00,
         1.09577025e+00,  8.64509929e-01],
       [ 4.99019249e+02, -4.62832583e+00,  2.37105336e+00,
         4.76533983e-01, -2.13214814e-01]])
```

3.0.1: Cumulative Variance Plot for PCA Model Optimization

Credit: https://365datascience.com/tutorials/python-tutorials/pca-k-means/ (https://365datascience.com/tutorials/python-tutorials/pca-k-means/)

In [32]:
```
# Plot to figure out how many components to use in actual PCA model
plt.figure(figsize = (10,8))
plt.plot(range(0,5), pca.explained_variance_ratio_.cumsum(), marker = "*",
plt.title("Cumulative Variance Explained by PCA Components")
plt.xlabel('# of PCA Components')
plt.ylabel('Cumualtive Explained Variance');

# Save fig
plt.savefig(save_path + '/CumVariancePCA_LINE.jpg')
```



3.0.1.1 Analysis:

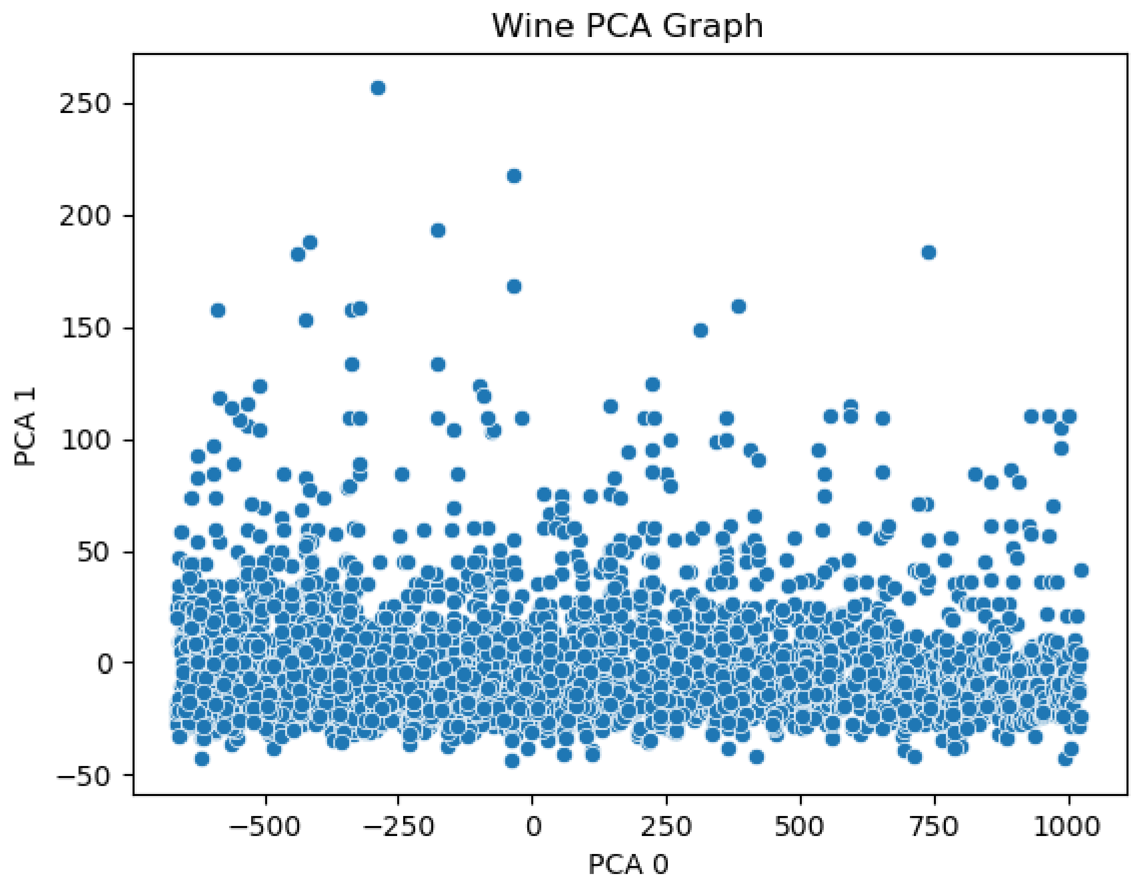
We can see from the cumulative explained variance graph above that after 2 components, there is a marginal increase in the percent of variance explained. Thus, to optimize the PCA model, we will use `n_components = 2`

In [33]:
```
pca = PCA(n_components=2, whiten=False)
x_pca = pca.fit_transform(x_vals)

# add features back into PCAdataframe (for plotting PCA)
df_mapped['PCA 0'] = x_pca[:, 0]
df_mapped['PCA 1'] = x_pca[:, 1]
```

In [34]:
```
# Seaborn scatter plot for display
sns.scatterplot(data=df_mapped, x="PCA 0", y="PCA 1").set(title='Wine PCA G

# Save fig
plt.savefig(save_path + '/PCAPlot_SCT.jpg')
```



In [35]:
```
# Interactive scatter plot and write to HTML graph
fig = px.scatter(df_mapped, x='PCA 0', y='PCA 1', hover_data=x_feat_list, t
fig.write_html('wine_mapped_PCA.html')
```

4.0: Machine Learning

4.0.1: K-Means Clustering Summary

Using an unsupervised clustering model, the aim is to be able to to cluster all wine features into "bins" to allow us to give a response back to the client on which wine they should select.

4.0.2: K-Means Clustering Model Optimization

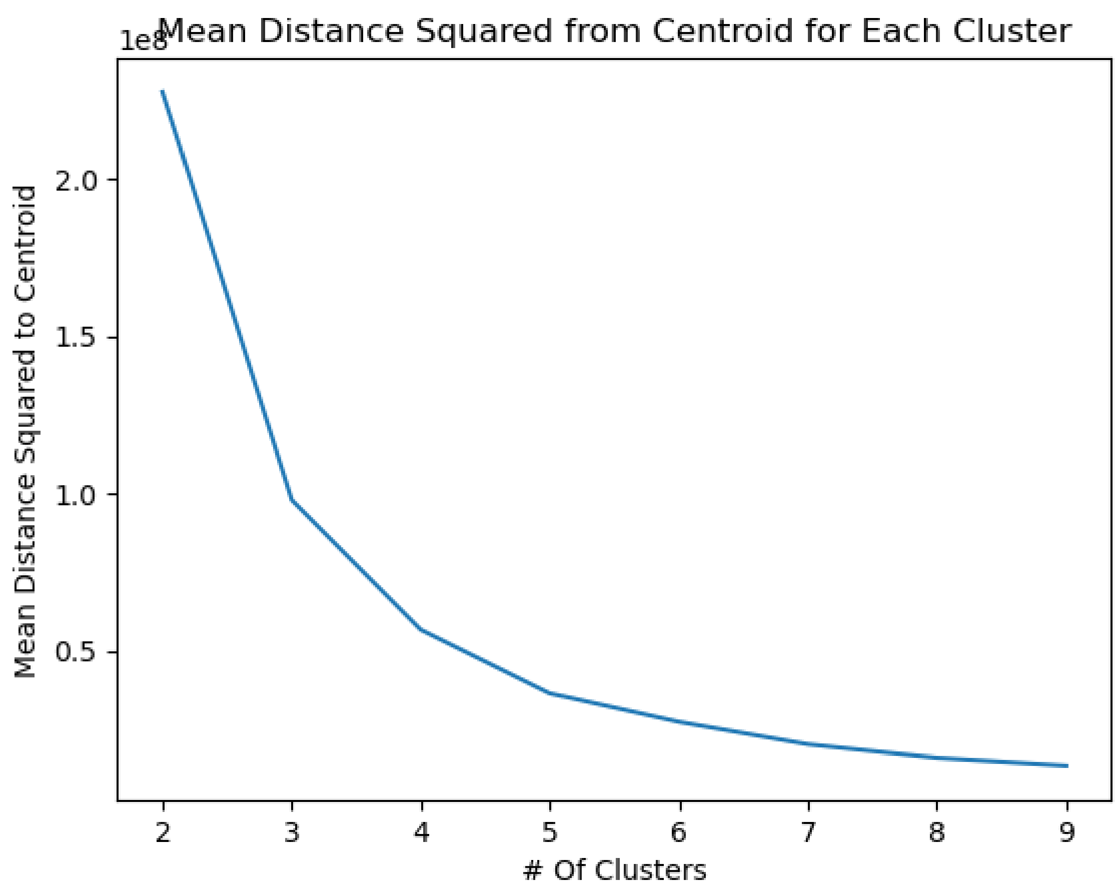
Code Credit: Professor Matt Higger at Northeastern University

In [36]:
```
# Optimize number of clusters
mean_d_dict = dict()
for n_clusters in range(2, 10):
    kmeans = KMeans(n_clusters=n_clusters)
    kmeans.fit(x_pca)
    y = kmeans.predict(x_pca)

    # compute & store mean distance
    mean_d = -kmeans.score(x_pca)
    mean_d_dict[n_clusters] = mean_d
```

In [37]:
```
# Graph mean distance to centroid to find the optimal n-value
plt.plot(mean_d_dict.keys(), mean_d_dict.values())
plt.xlabel('# Of Clusters')
plt.ylabel('Mean Distance Squared to Centroid');
plt.title('Mean Distance Squared from Centroid for Each Cluster');

# Save fig
plt.savefig(save_path + '/MeanD2Clusters_LINE.jpg')
```



4.0.3: Analysis:

We can see from the above graph comparing the mean distance squared from the centroid for each cluster decreases as the number of clusters increases. However, to optimize the K-Means Cluster algorithm, we must select the cluster at which the next decrease is marginal compared to the previous ones. Hence, `n_clusters=5` optimizes this.

4.1: Visualizing the K-Means Clustering

In [38]:
```
# X values are the features given in PCA y-axis
x = df_mapped['PCA 0'].values.reshape(-1,1)

# Fit in KMeans algorithm
kmeans = KMeans(n_clusters=5)
kmeans.fit_transform(x)
y = kmeans.predict(x)
```

In [39]:
```
sns.scatterplot(data=df_mapped, x='PCA 0', y='PCA 1', s=100, hue=y, palette
plt.gcf().set_size_inches(12, 6)
plt.title('K-Means Cluster Visualization')
plt.legend([f"Group {i}" for i in range(0,6)])

# Save fig
plt.savefig(save_path + '/ClusterChart_SCT.jpg')
```

In [40]:
```
df_mapped.head(5)
df_x = df_mapped[['processed_description']]
df_y = df_mapped[['variety']]
```

5.0: TFIDF

5.0.1: Bag-of-Words & TFIDF Vectorization

In [41]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vect = TfidfVectorizer()
bag_of_words = vect.fit_transform(df_x['processed_description'])
feature_names = vect.get_feature_names_out()
tfidf_df = pd.DataFrame(bag_of_words.toarray(), columns = feature_names)
tfidf_df.head(5)
```

Out[41]:

	000	02	03	04	05	06	08	09	10	100	...	zinfandel	zing	zingy	zins	zippy	zodiac	zo
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 6292 columns

5.0.2: PCA on TFIDF DataFrame

In [42]:
```
# X values are the features given in PCA y-axis
pca = PCA(n_components = 20)
features_standardized = StandardScaler().fit_transform(bag_of_words.toarray
reduced_data = pca.fit_transform(features_standardized)
```

6.0: ML for Predicting Wine Choice

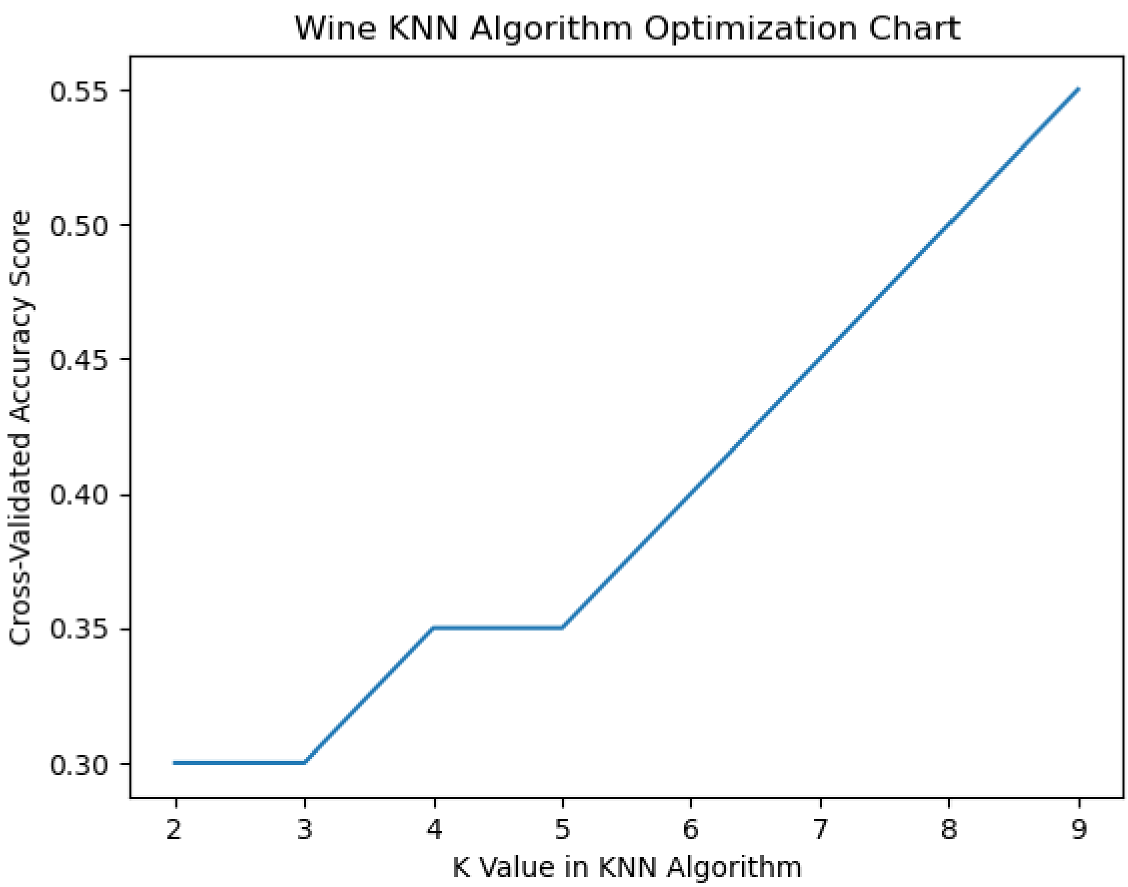
6.0.1: Data Splitting & Model Training

In [43]:
```
x_train, x_test, y_train, y_test = train_test_split(reduced_data, df_y['var
```

6.0.2: Parameter Tuning for K-Value

In [44]:
```
# Optimize for K 2-10
k_all = np.array(range(2, 10))
acc = np.empty(k_all.shape, dtype=float)
for idx, k in enumerate(k_all):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    acc[idx] = accuracy_score(y_test, y_pred)
```

In [45]:
```
# Plot optimization chart and save
plt.plot(k_all,acc)
plt.xlabel('K Value in KNN Algorithm');
plt.ylabel('Cross-Validated Accuracy Score');
plt.title('Wine KNN Algorithm Optimization Chart');
plt.savefig(save_path + '/KNNOptimizationChart_LINE.jpg')
```



6.0.2 Analysis

As seen as above, `n_neighbors = 3` is the optimal value to maximize performance of the KNN algorithm

6.0.3 KNN Algorithm Implementation

In [46]:
```
model = KNeighborsClassifier(n_neighbors=8)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

6.0.4 Metrics + Analysis

In [47]:
```
accuracy_score(y_test, y_pred)
```

Out[47]:
```
0.5
```

In [48]:
```
f1_score(y_test, y_pred,average='weighted')
```

Out[48]:
```
0.4541025641025641
```

From the constructed KNN algorithm with `n_neighbors = 3` produces a 40% accuracy score, meaning that 30% of the time, it predicts the right wine type according to the user's description of the wine. We also know that the model is able to classify 39.5% of all classifications in the dataset properly.

Despite not being hyper-accurate, this model is a good representation of real-life scenarios. No company will be able to accuractely pinpoint a singular wine variety for an emotion as there will always be more confounding variables. However, the model does give a start in the direction of the wine variety the consumer may like!