# Part 4 – Multiple Linear Regression

## I need to add sections on regularized regression, Ridge (L2), Lasso (L1), and Elastic net. Find source – Portilla?

## Multiple Linear Regression

**# Import Libraries**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Load Dataset**

**df = pd.read\_csv('Data/Advertising.csv')**

**df.head()**

**# MatpoltLib Scatter Plots - First Look at Relationships**

**fig,axes = plt.subplots(nrows=1,ncols=3,figsize=(16,6))**

**axes[0].plot(df['TV'],df['sales'],'o')**

**axes[0].set\_ylabel("Sales")**

**axes[0].set\_title("TV Spend")**

**axes[1].plot(df['radio'],df['sales'],'o')**

**axes[1].set\_ylabel("Sales")**

**axes[1].set\_title("Radio Spend")**

**axes[2].plot(df['newspaper'],df['sales'],'o')**

**axes[2].set\_ylabel("Sales")**

**axes[2].set\_title("Newspaper Spend")**

**plt.tight\_layout();**

**# A Piar Plot Diagram**

**sns.pairplot(df) ;**

**# Form X and y**

**X = df.drop('sales',axis=1)**

**y = df['sales']**

**# Train Test Split**

**from sklearn.model\_selection import train\_test\_split**

**# help(train\_test\_split)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=101)**

**X\_train**

**# Train Multiple Linear Regression Model**

**from sklearn.linear\_model import LinearRegression**

**# help(LinearRegression)**

**model = LinearRegression()**

**model.fit(X\_train, y\_train)**

**test\_predictions = model.predict(X\_test)**

**test\_predictions**

**# Model Evaluation**

**from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score**

**df['sales'].mean()**

**sns.histplot(data=df, x='sales')**

**mean\_absolute\_error(y\_test, test\_predictions)**

**mean\_squared\_error(y\_test, test\_predictions)**

**np.sqrt(mean\_squared\_error(y\_test, test\_predictions))**

**r2\_score(y\_test, test\_predictions)**

**# Deploy and use model**

**final\_model = LinearRegression()**

**final\_model.fit(X,y)**

**final\_model.coef\_**

**X.head()**

**from joblib import dump, load**

**dump(final\_model, 'final\_sales\_model.joblib')**

**loaded\_model = load('final\_sales\_model.joblib')**

**loaded\_model.coef\_**

**# New Prediction - 149 - TV, Radio - 22, Newspaper - 12**

**campaign = [[149, 22, 1]]**

**loaded\_model.predict(campaign)**

## Polynomial Regression

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**df = pd.read\_csv('Data/Advertising.csv')**

**df.head()**

**X = df.drop('sales', axis=1)**

**y = df['sales']**

**from sklearn.preprocessing import PolynomialFeatures**

**polynomial\_converter = PolynomialFeatures(degree=3, include\_bias=False)**

**polynomial\_converter.fit(X)**

**poly\_features = polynomial\_converter.transform(X)**

**poly\_features**

**poly\_features.shape**

**X.iloc[0]**

**poly\_features[0]**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(poly\_features, y, test\_size=0.3, random\_state=101)**

**from sklearn.linear\_model import LinearRegression**

**model = LinearRegression(fit\_intercept=True)**

**model.fit(X\_train, y\_train)**

**test\_predictions = model.predict(X\_test)**

**model.coef\_**

**from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error**

**MAE = mean\_absolute\_error(y\_test, test\_predictions)**

**MSE = mean\_squared\_error(y\_test, test\_predictions)**

**RMSE = np.sqrt(MSE)**

**# From Linear Regression MAE: 1.213 and RMSE 1.516 The Polynomial Model is performing much better**

**model.coef\_**

**# To compare coef\_'s**

**poly\_features[0]**

**X.iloc[0]**

**# CREATE THE DIFFERENCE ORDER POLYNOMIAL#**

**# split poly feature train/test**

**# fit on train**

**# store/save the rmse for both the train and test**

**# plot results**

**train\_rmse\_errors = []**

**test\_rmse\_errors = []**

**for d in range(1, 10):**

**poly\_converter = PolynomialFeatures(degree=d, include\_bias=False)**

**poly\_features = poly\_converter.fit\_transform(X)**

**# shift tab on train\_test\_split and copy example to ensure tuple is unpacked correctly**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(poly\_features, y, test\_size=0.3, random\_state=101)**

**model = LinearRegression(fit\_intercept=True)**

**model.fit(X\_train, y\_train)**

**train\_pred = model.predict(X\_train)**

**test\_pred = model.predict(X\_test)**

**train\_rmse = np.sqrt(mean\_squared\_error(y\_train, train\_pred))**

**test\_rmse = np.sqrt(mean\_squared\_error(y\_test, test\_pred))**

**train\_rmse\_errors.append(train\_rmse)**

**test\_rmse\_errors.append(test\_rmse)**

**train\_rmse\_errors**

**test\_rmse\_errors**

**plt.plot(range(1,6), train\_rmse\_errors[:5], label='Train RMSE')**

**plt.plot(range(1,6), test\_rmse\_errors[:5], label='Test RMSE')**

**plt.xlabel('Degree of Poly')**

**plt.ylabel('RMSE')**

**plt.legend()**

## Final Model Deployment

**final\_poly\_converter = PolynomialFeatures(degree=3, include\_bias=False)**

**final\_model = LinearRegression()**

**full\_converted\_X = final\_poly\_converter.fit\_transform(X)**

**final\_model.fit(full\_converted\_X,y)**

**from joblib import dump, load**

**dump(final\_model, 'final\_poly\_model.joblib')**

**dump(final\_poly\_converter, 'final\_converter.joblib')**

**loaded\_converter = load('final\_converter.joblib')**

**loaded\_model = load('final\_poly\_model.joblib')**

**campaign = [[149, 22, 12]]**

**transformed\_data = loaded\_converter.fit\_transform(campaign)**

**loaded\_model.predict(transformed\_data)**