Aryan Patel - Analytics of Business Intelligence Project Report
For this report I will show code for each question and the results for each model and discuss.

I) (Preprocessing) code : `#sklearn preprocessing on training data`

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder, FunctionTransformer
from sklearn.compose import make_column_selector as selector

X_train = training_data.drop(columns=['SalePrice'])
y_train = training_data['SalePrice']


X_test = test_data.drop(columns=['SalePrice'])
y_test = test_data['SalePrice']





numerical_features = selector(dtype_include=np.int64)
categorical_features = selector(dtype_include=object)


numerical_transformer = Pipeline(steps=[
('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
('onehot', OneHotEncoder())
])

preprocessor = ColumnTransformer(
transformers=[
('num', numerical_transformer, numerical_features),
('cat', categorical_transformer, categorical_features)
])


pipeline = Pipeline(steps=[
('preprocessor', preprocessor)
])
```

```
X_train_processed = pipeline.fit_transform(X_train)


X_test_processed = pipeline.transform(X_test)


y_train_processed = np.log1p(y_train)
y_test_processed = np.log1p(y_test)
```

This code will standardize the X variables that are int and will one hot encode the catgeorical variables. Also it is splitting into X and Y train . As data already came with test and train split we just needed to drop Y to get X data and only select Y for Y data for both test and train. We also took the log of y as requested (Log of Sales Price). No null or Na values so not much processing was needed.

#II Simple multiple linear regression(no regularization)

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error


linear_regression = LinearRegression()


linear_regression.fit(X_train_processed, y_train_processed)


y_train_pred = linear_regression.predict(X_train_processed)
y_test_pred = linear_regression.predict(X_test_processed)


train_mse = mean_squared_error(y_train_processed, y_train_pred)
test_mse = mean_squared_error(y_test_processed, y_test_pred)

print("Training MSE (Regular Regression) is :", train_mse)
print("Test MSE (Regular Regression) is :", test_mse)
```

```
residual_model0_test = y_test_processed - y_test_pred
```

As question II asked we made a simple multiple linear regression model and were able to calculate the test /training mse of : Training MSE (Regular Regression) is : 0.02462790457299441 Test MSE (Regular Regression) is : 0.064289085860186. Data is standardized so sales price is not exactly 0.02 MSE of Test but we can see that it is already quite accurate as a base model.

III. Elastic Net regression model code : `from sklearn.linear_model import ElasticNet`

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

#tried many values for all params but this resulted in best from my testing
param_grid = {
'alpha': [ 0.1, 0.5, 1.0, 10, 100],
'l1_ratio': [ 0.1, 0.5, 0.9, 0.99]
}



elastic_net = ElasticNet(max_iter=10000)



grid_search = GridSearchCV(elastic_net, param_grid, cv=10,
scoring='neg_mean_squared_error')

grid_search.fit(X_train_processed, y_train_processed)



best_elastic_net = grid_search.best_estimator_



print("Best Hyperparameters are :")
print(grid_search.best_params_)



best_elastic_net.fit(X_train_processed, y_train_processed)



y_train_pred = best_elastic_net.predict(X_train_processed)
```

```
y_test_pred = best_elastic_net.predict(X_test_processed)


train_mse_elastic = mean_squared_error(y_train_processed, y_train_pred)
test_mse_elastic = mean_squared_error(y_test_processed, y_test_pred)

print("Training MSE (Elastic Net) is :", train_mse_elastic)
print("Test MSE (Elastic Net) is :", test_mse_elastic)

residual_model1_test = y_test_processed - y_test_pred
```

Results of the Elastic Net model were : Best Hyperparameters are : {'alpha': 0.1, 'l1_ratio': 0.1} Training MSE (Elastic Net) is : 0.027969114268482042 Test MSE (Elastic Net) is : 0.05836975904396832. As we can see the elastic net model performs better than the regular multiple regression slightly on test mse which makes sense as regularization helps with overfitting. Even though the test accuracy is slightly worse then base multiple regression by 0.03 mse the test mse is improved.And since we are concerned with how model does to unseen data this is better then regular regression model. Especially with as few data as we have it would make sense to introduce regularization so we do not overfit too much to the little data that we have for training. We also used grid search to find the best alpha and l1 ratio with cross validation. Elastic net performs better than base regression most likely because we have little data and that helps prevent the regression model from overfitting on our little amount of data.

IV) SVM Model code :

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

#tried large range of param but these resulted in best mse from what I have tested.
Also tested linear/poly svm and different degrees. Poly svm with degree 3 had best
results.
param_grid = {
'C': [0.01,0.1, 1, 10],
'epsilon': [0.01,0.1, 0.5, 1.0]
}



svm_model = SVR(kernel='poly',degree=3)
```

```
grid_search = GridSearchCV(svm_model, param_grid, cv=5,
scoring='neg_mean_squared_error')
grid_search.fit(X_train_processed, y_train_processed)


best_svm_model = grid_search.best_estimator_


best_svm_model.fit(X_train_processed, y_train_processed)


y_train_pred_svm = best_svm_model.predict(X_train_processed)
y_test_pred_svm = best_svm_model.predict(X_test_processed)


train_mse_svm = mean_squared_error(y_train_processed, y_train_pred_svm)
test_mse_svm = mean_squared_error(y_test_processed, y_test_pred_svm)

print("Training MSE (SVM):", train_mse_svm)
print("Test MSE (SVM):", test_mse_svm)
print("Best hyperparameters:", grid_search.best_params_)

residual_model2_test = y_test_processed - y_test_pred_svm
```

SVm model results/discussion : Training MSE (SVM): 0.009404704464279212 Test MSE (SVM): 0.028397274500036488 Best hyperparameters: {'C': 1, 'epsilon': 0.1}. The SVm model results in the best test mse accuracy out of all models. I tried linear and poly kernel and different degrees for the poly kernel but the best I found for test mse was poly and degree 3 for the svm model. This resulted in very good mse score of 0.0283 for test which is very accurate in predicting Sales Price. SVm also used cross validation with grid search to find best parameter values.

V) Neural Network Model

Code for neural network model :

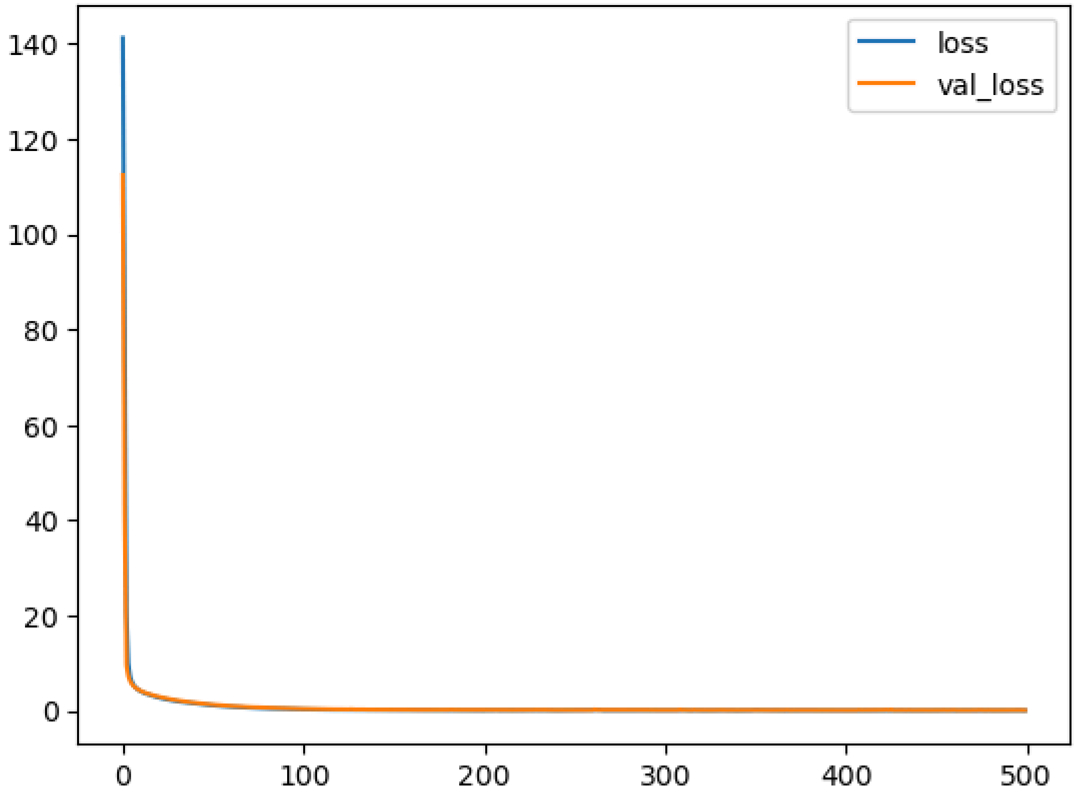
```
from sklearn.model_selection import train_test_split


# Creating validation data for neural network
```

```
X_train_nn, X_valid_nn, y_train_nn, y_valid_nn = train_test_split(X_train_processed,
y_train_processed, random_state=42)

x_train_processed_view = X_train_processed.shape

print(x_train_processed_view)
import tensorflow as tf




tf.keras.backend.clear_session()
tf.random.set_seed(42)

ann_model = tf.keras.Sequential([
tf.keras.layers.Dense(70, activation="relu", input_shape=(28,)),
tf.keras.layers.Dense(35, activation="relu", kernel_initializer="he_normal",
kernel_regularizer=tf.keras.regularizers.l2(0.05)),
tf.keras.layers.Dropout(0.001),
tf.keras.layers.Dense(20, activation="relu", kernel_initializer="he_normal",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.Dense(1)
])



ann_model.summary()
ann_model.compile(loss='mean_squared_error', optimizer='adam')
history = ann_model.fit(X_train_nn, y_train_nn, epochs=500,
validation_data=(X_valid_nn, y_valid_nn),
)
y_test_pred = ann_model.predict(X_test_processed)
nnmse = mean_squared_error(y_test_processed, y_test_pred, squared=True)

y_test_pred1 = ann_model.predict(X_test_processed).ravel()



y_test_processed1 = y_test_processed.ravel()

residual_nn_test = y_test_processed1 - y_test_pred1
print("Test MSE (NN):", nnmse)

pd.DataFrame(history.history).plot()
```

Neural Network results/ discussion : 8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step 8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step Test MSE (NN): 0.04430107075982362. As we can see the neural network had test mse of 0.04 which would make it the second best model to use for predicting Sales Price based on Test MSE. For the neural network I used many different layers and number of neurons but based on the low data that we had I ended up using this

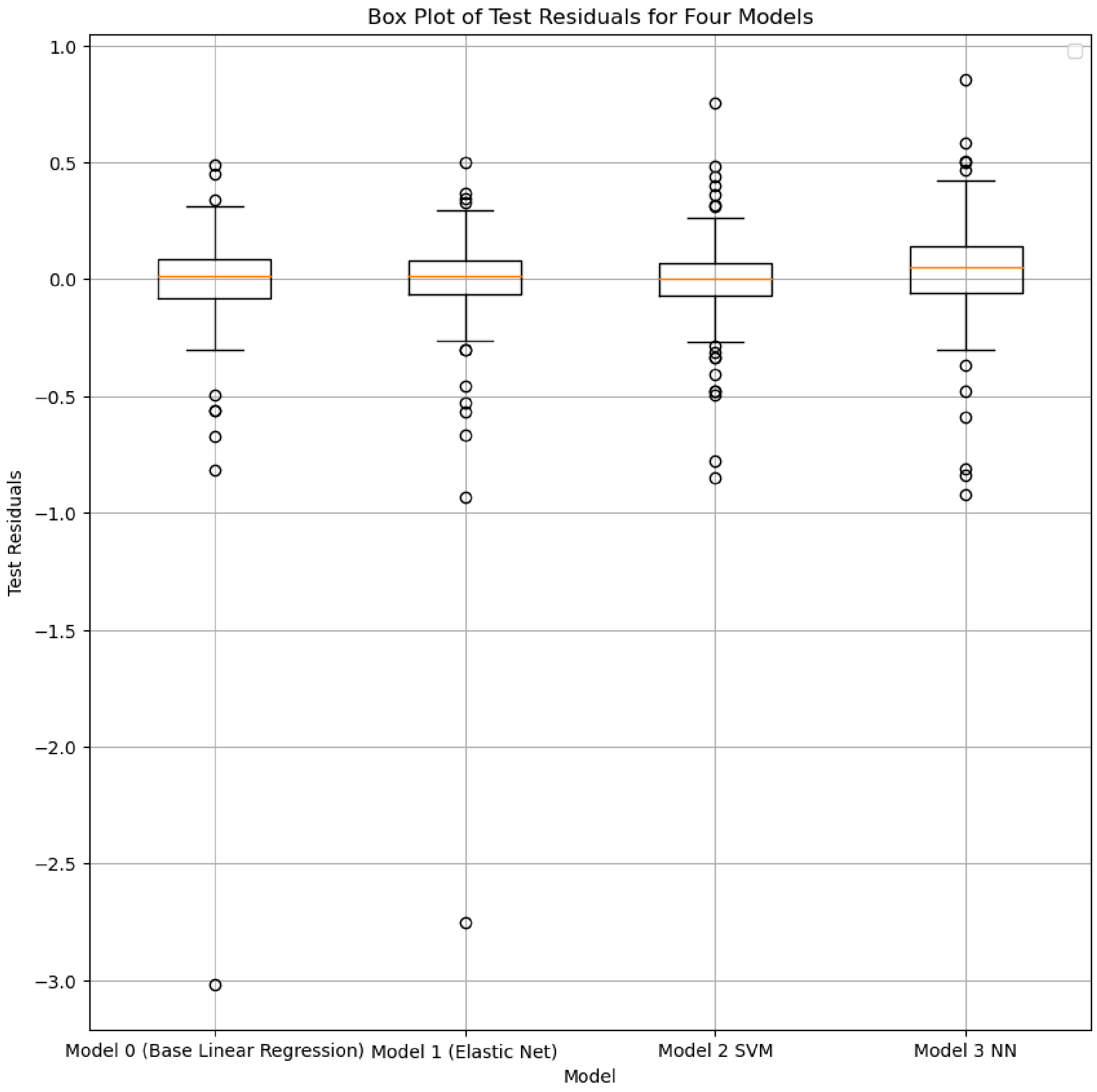
```
ann_model = tf.keras.Sequential([
tf.keras.layers.Dense(80, activation="relu", input_shape=(28,)),
tf.keras.layers.Dense(40, activation="relu", kernel_initializer="he_normal",
kernel_regularizer=tf.keras.regularizers.l2(0.05)),
tf.keras.layers.Dropout(0.001),
tf.keras.layers.Dense(20, activation="relu", kernel_initializer="he_normal",
kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.Dense(1)
])
```

As we can see i have introduced regularization to reduce overfitting on test data. I used a kernel intializer for faster optimization at each layer. I also included a very small dropout which i found to product the best test mse. Input layer is 28 as final x train processed has 28 columns. Final output layer is 1 as it is regression and returning 1 number. Choose activation relu again for fast optimization and as since sales price is positive from 0-infinity relu is a good choice. Also at the beginning of the code chunk for this nn you can see I created validation data by splitting the full training data again this is due to neural networks requiring validation data to be accurate as it can overfit very easily with the large amount of param it has compared to regression.



This is the learning curve showing loss vs val loss through the epocs. We can see the mosel is slightly underfit as the val loss is above the loss very slightly. But this is an ideal model based on learning curves shown above as both the val loss and loss are very close together.

VI) Discussing final model performance of the four models . final model performance from the base regression model, elastic net regression, svm, and neural network. I collected the residuals of each model and plotted box and whisker plot as requested for the test data.



As we can see from the plot of the four models test residuals we can see the SVM model most accurately predicts the test mse as it has the least deviation from the mean of 0 it also has fewer outliers and has the tightest gap between quartiles compared to all models this means that all the predictions that the svm makes is very accurate. We also do not have many outliers like the other models, especially the base regression or elastic net which produced some of the largest outliers. Also the SVM is the only model with median 0 which is what we want from an model. All of the other models are slightly off showing some bias either towards over or underpredicting. We can see that the neural network comes in second in terms of prediction accuracy on test mse by showing the collected residuals of the four models shown by the tight residual spread. After neural network is Elastic net then finally our base regression model. The neural network used around 500 epochs and I found that using early stopping or decreasing epochs decreased the test accuracy so it took around 1 min to train. The svm only took 2.3 seconds and was more accurate than the neural network. The neural network slightly changed test mse each time it was ran even though random seeds were set, but consisetnely it was around the second best model. All other models can be trained multiple times with same code but will give same output. Possibly the other forms of neural networks beside our feed forward mlp that was taught in class could be more accurate. But as of now the SVM model is fast computationally and the most accurate in terms of test accuracy comparing all the four models.

Best model : SVM (with poly kernel and degree = 3), based on test mse and residual analysis of test mse of the four models.
