Summary for the CEO:

General trend:

Price vs. Year:

There is significant **negative** correlation between **year** and **price**. The older the car the lower the price. Price of the car **decreases** by average **$425** per year.

Price vs Odometer:

There is significant **negative** correlation between **odometer** and **price**. The more the car has run on the road the lower the price. Price of the car decreases by average **0.0478 cents per mile**. So, if the car has run **1000** miles the price will decrease **by $47.8.**

Price vs. Manufacturer:

Can't state clear as Ford is the most popular in provided dataset. Reasoning based on this can introduce bias

Price vs. Colour:

We can observe increase in the used car price based on colour. **Black**, **blue** and **red** cars are slightly more expensive than the rest. On average **black** cars are around **$1500** more expensive, while **red** and **blue** cars are around **$1100** more expensive than other colours.

Price vs. Type:

Trucks and **pickups** are more expensive than other types of cars. On average trucks and pickups are **$7000** more expensive than other types of cars.

Price vs. Transmission:

Automatic transmission cars are more expensive than manual ones. On average they are **$1000** more expensive than manual ones.

Price vs Cylinders:

4-cylinder cars are more expensive than **6-cylinder cars**. On average 4-cylinder cars are **$200 more** expensive than **6-cylinder cars**. **8-cylinder cars** are the most expensive ones. On average they are **$1400** more expensive than 4-cylinder cars.

Price vs Condition:

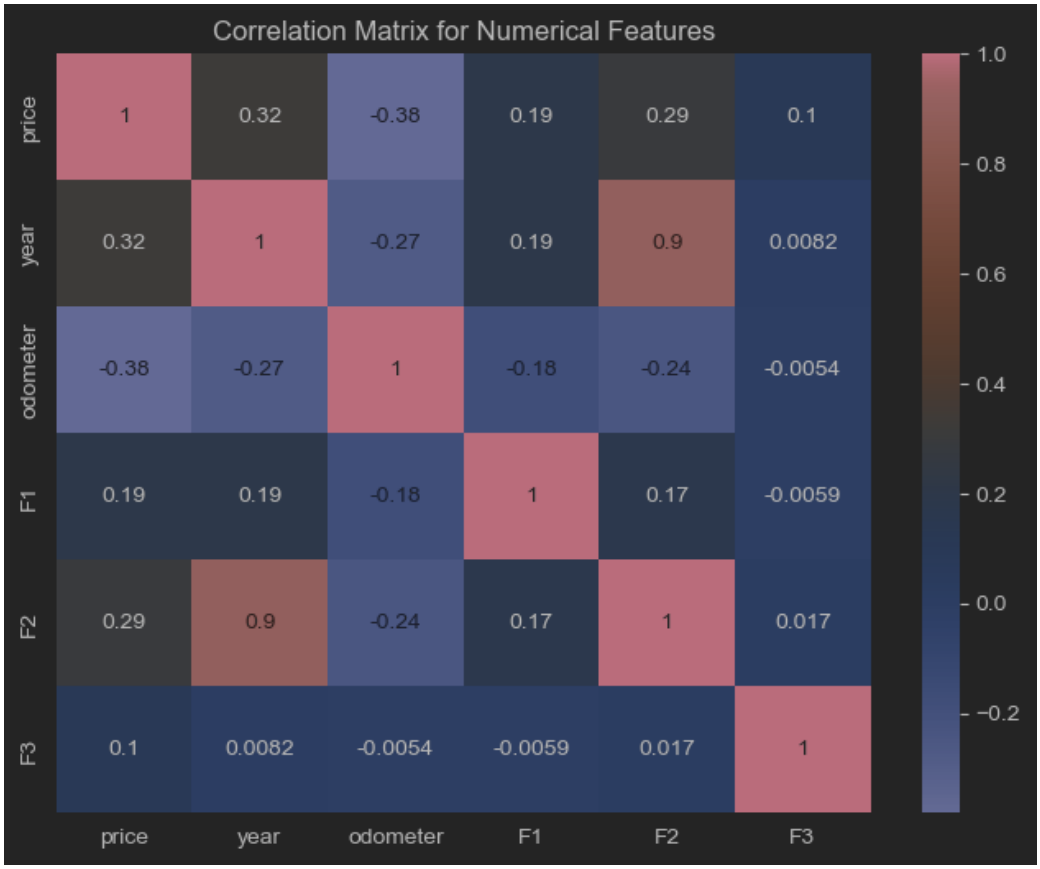
It's best to sell a car with **'excellent'** or **'like new' condition**. On average cars in **'excellent'** condition are $3000 more expensive than cars in **'good'** condition. Cars in **'like new'** condition are $6000 more expensive than cars in **'good'** condition.

F1, F2, F3 and F4 impact on price

F1 and F3 factors have positive impact on price. F2 is strongly correlated to the year of the car and has small impact on price once isolated. F4 factor has negligible impact on price.

A technical discussion for the technical manager

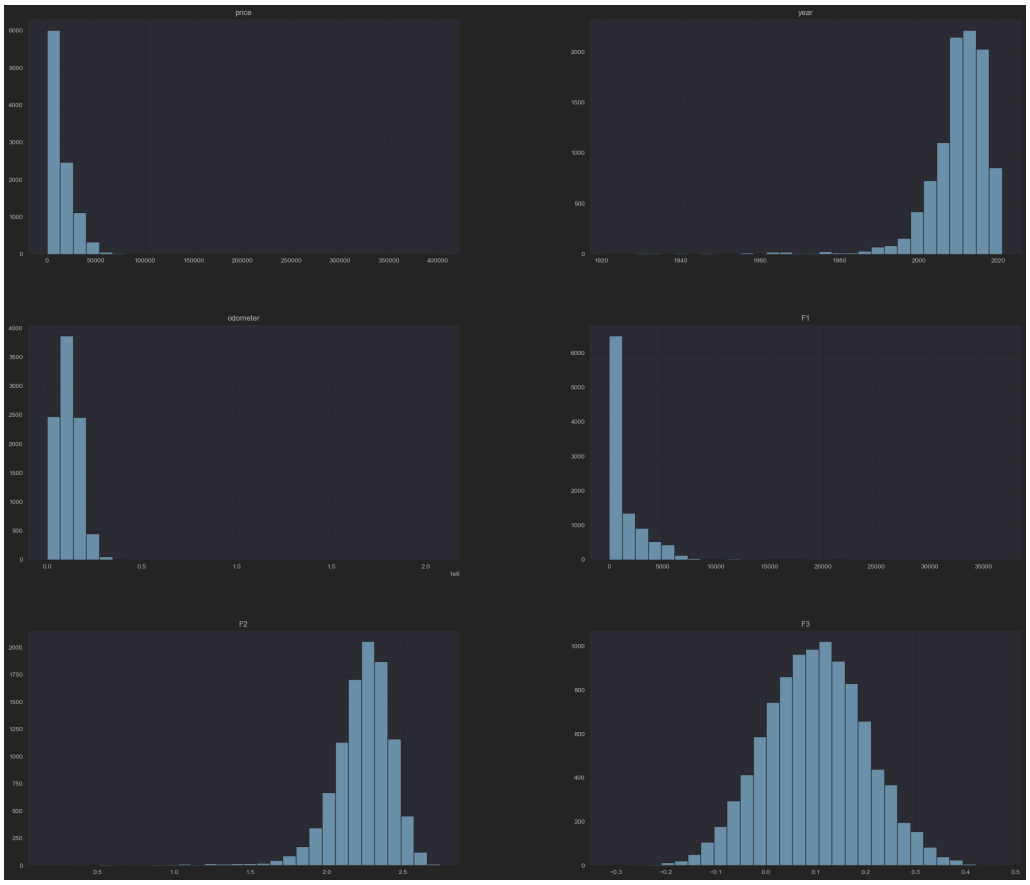
After loading the data, I performed initial analysis of dataset features. We could identify 8 categorical columns from which **'fuel'** could be deleted, because contained only one type: **gas**. From correlation matrix I could conclude which features are good to explore at first shot:



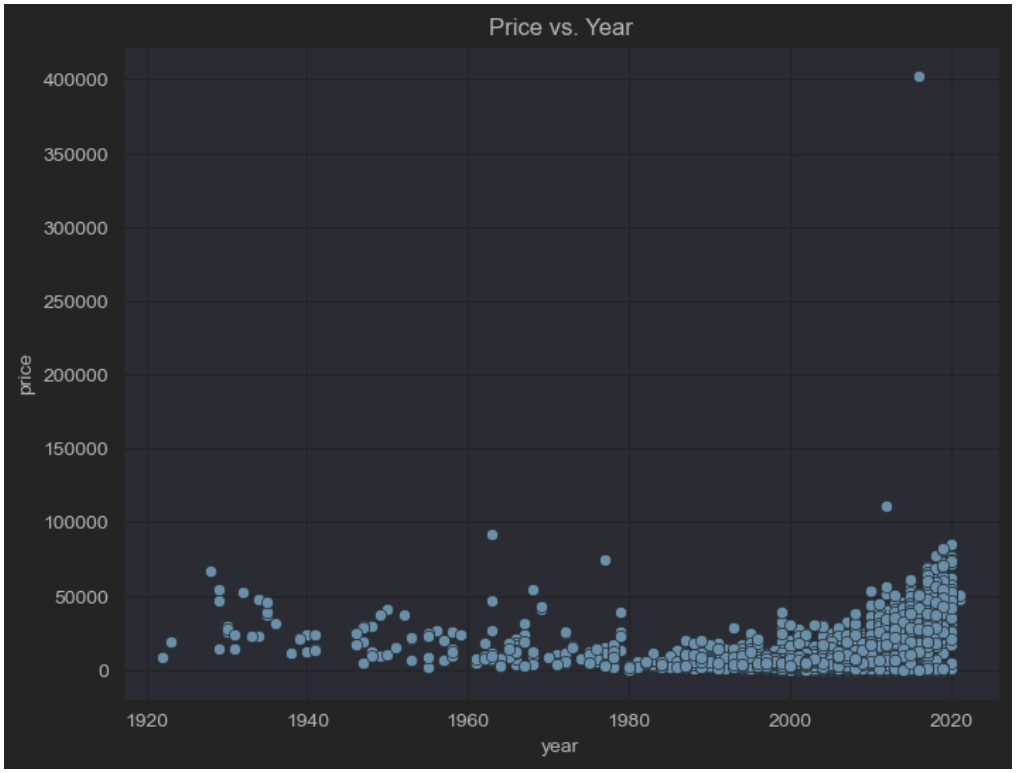
At the first glance we can see pretty high (as for that dataset), positive correlation between price and year, F1, F2 features.

Further analysis showed up that F2 strong correlation with year was impacting "standalone" once isolated.

Simple histograms of numerical columns described how data is distributed. We can see that only F3 was assembling normal distribution. Other features need preprocessing before training the model.



Further plotting showed up that we have quite high outlier in price data:



And I simply decided to remove it.

Plotting almost all features was simple method to have a basic understanding how they impact price. Having that I was ready to prepare my dataset for model training.
Clear dataset was read into a dataframe, and as first step one-hot-encoded all categorical features. Columns **'odometer', 'F1', 'F2', 'F3'** were scaled using sklearn StandardScaler as their value was highly scattered. After that we could split the data into three sets: training, test and validation, and I could start training desired models.

The best model was selected based on RMSE value. The one with the lowest one wins. Below you can see validation results:

LinearRegression: 9004.45

Ridge_alpha_0.001: 9004.45

Ridge_alpha_0.01: 9004.45

Ridge_alpha_0.1: 9004.47

Ridge_alpha_1: 9004.72

Ridge_alpha_10: 9007.22

Ridge_alpha_100: 9034.19

Lasso_alpha_0.001: 9004.45

Lasso_alpha_0.01: 9004.45

Lasso_alpha_0.1: 9004.49

Lasso_alpha_1: 9004.92

Lasso_alpha_10: 9009.33

Lasso_alpha_100: 9078.24

ElasticNet_alpha_0.001: 9005.21

ElasticNet_alpha_0.01: 9012.43

ElasticNet_alpha_0.1: 9085.38

ElasticNet_alpha_1: 9527.82

ElasticNet_alpha_10: 10502.07

ElasticNet_alpha_100: 11115.59

Best model accuracy and error on test set shows as follows:

Test RMSE: 9631.48

Model accuracy: 36.92%

Results are quite poor, but we were limited only to linear models. We can proceed with further feature analysis, so they should increase.

Print-out of the code for the senior developer

We read the data and preprocess it. Encode categorical columns, drop columns redundant in model training, fill missing values, or remove such samples from dataset:

```
df = pd.read_csv('data/used_car_dataset.csv')
✓ [58] 15ms

labelEncoder = LabelEncoder()

df['manufacturer'] = labelEncoder.fit_transform(df['manufacturer'])
df['paint_color'] = labelEncoder.fit_transform(df['paint_color'])
df['transmission'] = labelEncoder.fit_transform(df['transmission'])
df['type'] = labelEncoder.fit_transform(df['type'])
df['cylinders'] = labelEncoder.fit_transform(df['cylinders'])
df['condition'] = labelEncoder.fit_transform(df['condition'])
df['F4'] = labelEncoder.fit_transform(df['F4'])
✓ [59] 15ms

df.drop(columns=['fuel'], inplace=True)
df = df.loc[df['price'] < 200000]
df.dropna(inplace=True)
df['year'] = df['year'].astype(int)
df['odometer'] = df['odometer'].astype(int)
✓ [60] < 10 ms
```

Next, we split our data on features and target:

```
X = df.drop('price', axis=1)
y = df['price']

features_to_scale = ['odometer', 'F1', 'F2', 'F3']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()

X_train[features_to_scale] = scaler.fit_transform(X_train[features_to_scale])
X_test[features_to_scale] = scaler.transform(X_test[features_to_scale])

X_train_sub, X_val, y_train_sub, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=42)
✓ [61] 13ms
```

Setup variables for searching for the best model:

```
alphas = [0.001, 0.01, 0.1, 1, 10, 100]     alphas

results = {}
best_model = None
best_model_name = None
lowest_rmse = float('inf')
✓ [62] < 10 ms
```

Start training process, looking for the best model in assumed linear ones:

```
lr = LinearRegression()
lr.fit(X_train_sub, y_train_sub)
y_val_pred = lr.predict(X_val)
rmse_lr = np.sqrt(mean_squared_error(y_val, y_val_pred))
results['LinearRegression'] = rmse_lr

if rmse_lr < lowest_rmse:
    lowest_rmse = rmse_lr
    best_model = lr
    best_model_name = 'LinearRegression'

for model_name, model_class in [('Ridge', Ridge), ('Lasso', Lasso), ('ElasticNet', ElasticNet)]:
    for alpha in alphas:
        model = model_class(alpha=alpha, random_state=42)
        model.fit(X_train_sub, y_train_sub)
        y_val_pred = model.predict(X_val)
        rmse = np.sqrt(mean_squared_error(y_val, y_val_pred))
        results[f"{model_name}_alpha_{alpha}"] = rmse
        if rmse < lowest_rmse:
            lowest_rmse = rmse
            best_model = model
            best_model_name = f"{model_name}_alpha_{alpha}"
✓ [65] 128ms
```

The best one is selected based on the lowest mean squared error value.

We can print the results:

```
print("Validation RMSE results:")
for key, value in results.items():
    print(f"{key}: {value:.2f}")
✓ [66] < 10 ms

Validation RMSE results:
LinearRegression: 9004.45
Ridge_alpha_0.001: 9004.45
Ridge_alpha_0.01: 9004.45
Ridge_alpha_0.1: 9004.47
Ridge_alpha_1: 9004.72
Ridge_alpha_10: 9007.22
Ridge_alpha_100: 9034.19
Lasso_alpha_0.001: 9004.45
Lasso_alpha_0.01: 9004.45
Lasso_alpha_0.1: 9004.49
Lasso_alpha_1: 9004.92
Lasso_alpha_10: 9009.33
```

And finally test our model on validation set:

```
print(f"\nBest model on validation set: {best_model_name} with RMSE = {lowest_rmse:.2f}")

X_train_val = pd.concat([X_train_sub, X_val])
y_train_val = pd.concat([y_train_sub, y_val])
best_model.fit(X_train_val, y_train_val)

y_test_pred = best_model.predict(X_test)
test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
print(f"\nTest RMSE: {test_rmse:.2f}")
print(f"Model accuracy: {best_model.score(X_test, y_test) * 100:.2f}%")
✓ [64] 22ms

Best model on validation set: LinearRegression with RMSE = 9004.45
Best model on validation set: LinearRegression with RMSE = 9004.45

Test RMSE: 9631.48
Model accuracy: 36.92%
```

{} Code M↓ Markdown

RMSE and model accuracy are quite unsatisfactory, so we should proceed with further feature analysis, selection or maybe even data augmentation as provided dataset has low sample count.[1]

[1] Provided code samples were partially generated with help of ChatGPT o3-mini-high model