Why Predicting Delivery Time Matters

Behind every successful food delivery is a *clock ticking*.

On-time delivery

Late delivery

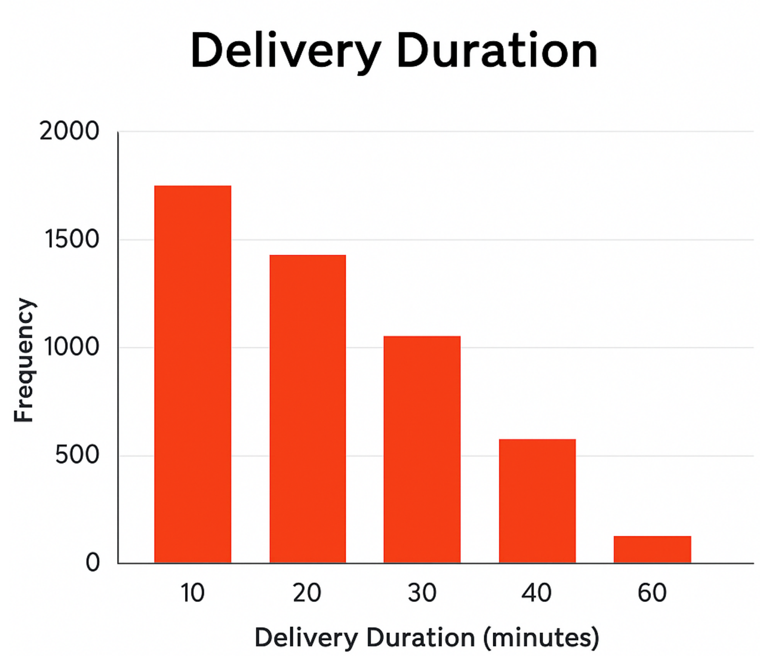
Customers remember late deliveries, not the algorithm.

That's why companies like DoorDash invest heavily in predicting delivery time.

My Goal: Predict delivery time using 170,000+ DoorDash logs.

170,000+ Orders, One Goal: Accuracy

Delivery Duration



Sample Data

market_id	created_at	actual_delivery_time	store_id	store_primar	order_protoc	total_items	subtotal	num_distinct	min_item_pri	max_item_pr
1	2/6/15 22:24	2/6/15 23:27	1845	american	1	4	3441	4	557	1239
2	2/10/15 21:49	2/10/15 22:56	5477	mexican	2	1	1900	1	1400	1400
3	1/22/15 20:39	1/22/15 21:09	5477	NA	1	1	1900	1	1900	1900
3	2/3/15 21:21	2/3/15 22:13	5477	NA	1	6	6900	5	600	1800
3	2/15/15 2:40	2/15/15 3:20	5477	NA	1	3	3900	3	1100	1600
3	1/28/15 20:30	1/28/15 21:08	5477	NA	1	3	5000	3	1500	1900
3	1/31/15 2:16	1/31/15 2:43	5477	NA	1	2	3900	2	1200	2700
3	2/12/15 3:03	2/12/15 3:36	5477	NA	1	4	4850	4	750	1800
2	2/16/15 0:11	2/16/15 0:38	5477	indian	3	4	4771	3	820	1604
3	2/18/15 1:15	2/18/15 2:08	5477	NA	1	2	2100	2	700	1200
3	2/2/15 19:22	2/2/15 20:09	5477	NA	4	4	4300	4	1200	1500
3	2/16/15 4:19	2/16/15 6:34	5477	NA	1	2	2200	2	600	1600
3	2/7/15 1:34	2/7/15 2:17	5477	NA	1	1	1900	1	1900	1900
3	1/25/15 1:50	1/25/15 2:28	5477	NA	4	4	4986	4	699	2362
1	2/12/15 3:36	2/12/15 4:14	2841	italian	1	1	1525	1	1525	1525
1	1/27/15 2:12	1/27/15 3:02	2841	italian	1	2	3620	2	1425	2195

- **Dataset:** 170,000+ DoorDash delivery logs from early 2015.
- **Target Variable:** total seconds from order creation to delivery.
- **Key Fields:** full timestamps, market and order attributes, driving time estimates

Turning Raw Data into Real Insights

New Features Created

```python
# Average price per time:
train_df['avg_price_per_item'] = train_df['subtotal'] / train_df['total_items']

# Item diversity ratio:
train_df['item_diversity_ratio'] = train_df['num_distinct_items'] / train_df['total_items']

# Price range of items:
train_df['price_range_of_items'] = train_df['max_item_price'] - train_df['min_item_price']
```

Encoded Categorical Variables

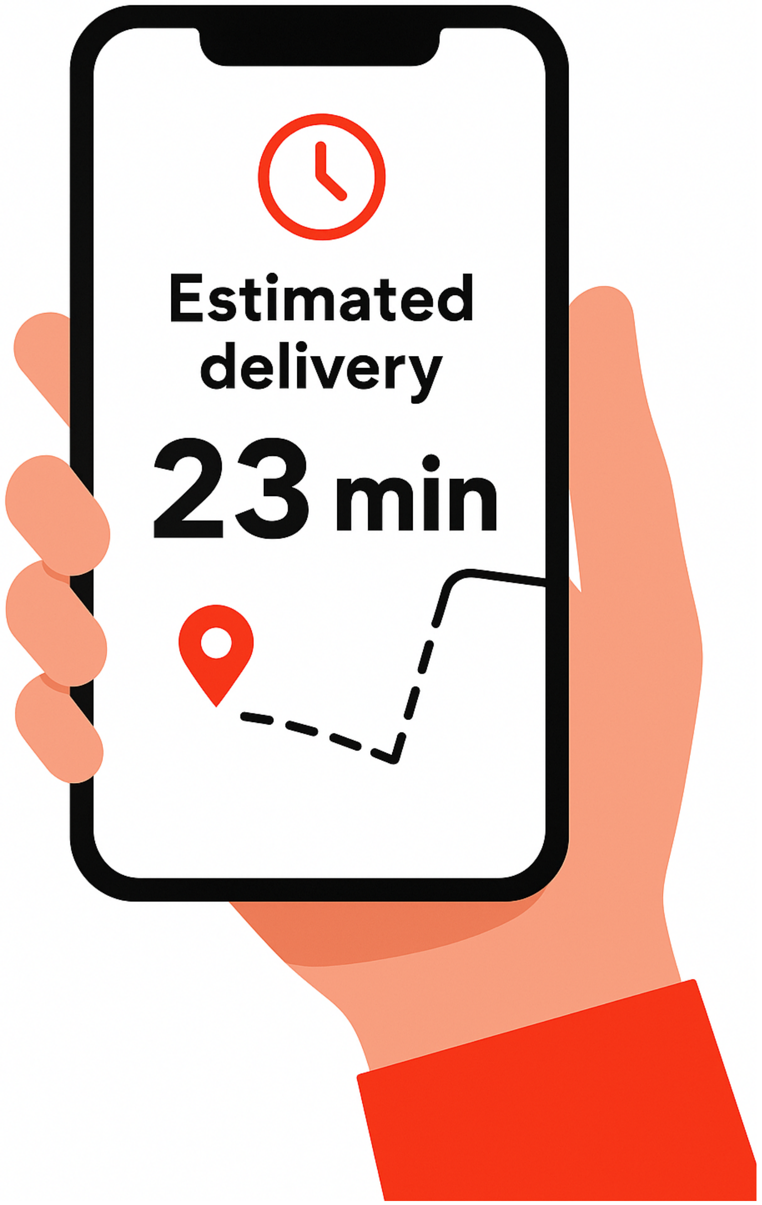
```python
# Encode low cardinality categorical variables:
# One-hot encode order protocol and prefix with identifier
order_protocol_dummies = pd.get_dummies(df.order_protocol).add_prefix('order_protocol_')

# One-hot encode market_id and prefix with identifier
market_id_dummies = pd.get_dummies(df.market_id).add_prefix('market_id_')
```

```python
# Impute and encode store_primary_category:
# Get a list of all unique store IDs
store_id_unique = df["store_id"].unique().tolist()

# Build a mapping of store_id to the most common primary category (mode)
store_id_and_category = {
    store_id: df[df.store_id == store_id].store_primary_category.mode().iloc[0]
    for store_id in store_id_unique
    if not df[df.store_id == store_id].store_primary_category.mode().empty
}
```

What Drives Delivery Time



Top Features

- Estimated driving time
- Average price per item
- Total outstanding orders
- Busy ratio (i.e. how busy all the drivers get)

Models Compared

- Linear Regression
- Ridge, Lasso, and PLS
- Decision Tree
- Random Forest
- XGBoost

Feature Selection

- Correlation heat-maps
- VIF
- Recursive feature elimination

Evaluation Metric

- Root Mean Squared Error (RMSE)

Key Takeaways

Findings

- Estimated driving time and market congestion are top drivers.
- Regularization improves interpretability.

Applications

- Accurate ETAs reduce cancellations.
- Real-time dasher data enables better logistics.

Learnings

- Feature engineering > complex models
- Interpretability often beats marginal accuracy gains.

Author : Satkar Karki